Attorney Docket No. **09792909-5269** PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Patent Application of:	)	Group Art Unit: **2665**
	)	
Kazuhiro Shitama	)	Examiner: **Cynthia L. Davis**
	)	
Application No. **10/045,320**	)	
	)	
Filed: **November 9, 2001**	)	
	)	
For: NETWORK CONNECTION CONTROL APPARATUS AND METHOD	)	

MAIL STOP APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPELLANT'S BRIEF ON APPEAL

Dear Sir:

In accordance with the provisions of 37 C.F.R. § 41.37, Appellant herewith submits this Brief in support of the Appeal for the above-referenced application.

I. REAL PARTY IN INTEREST

The real party in interest in the present appeal is the Assignee, Sony Corporation, a Japanese Corporation. The Assignment was recorded in the U.S. Patent and Trademark Office at Reel 012818, Frame 0493 on April 15, 2002.

II. RELATED APPEALS AND INTERFERENCES

There are no related appeals and no related interferences.

III. STATUS OF CLAIMS

Claims 1-9 are pending in this application. The present Appeal is directed to claims 1-9 that were rejected under 35 U.S.C. § 103(a) as being unpatentable over Fan (U.S. Patent No. 6,219,706) in view of Abadi (U.S. Patent No. 5,315,657) in a final office action dated December 27, 2005.

IV. STATUS OF AMENDMENTS

There are no pending amendments. However, Appellant reserves the right to submit an amendment to correct noted typographical errors that do not affect the appeal.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The invention is directed to a network connection control apparatus and method for granting or rejecting access when a device on a global network demands access to services provided on a local network. (See page 8, line 10 through page 9, line 21.) The method comprises the steps of authenticating the device on the global network in response to a service access request message (see page 14, lines 2-12 and steps S1 and S2 in Fig. 3); creating an access permission entry in response to an access request from the authenticating device (see page 14, line 16 through page 15, line 9 and step S4 in Fig. 3) and adding the access permission entry to an access permission list (see page 15, lines 10-11 and step 5 in Fig. 3); and determining, upon receiving a data packet from a device on the global network, whether or not the data packet should be transferred to the local network based on information extracted from the header of the

data packet and on the access permission entry contained in the access permission list (see page 16, line 11 through page 17, line 11 and steps SS1 and SS2 in Fig. 5).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 1-9 stand rejected under 35 U.S.C. § 103(a) as obvious over Fan (U.S. Patent No. 6,219,706) in view of Abadi (U.S. Patent No. 5,315,657).

VII. ARGUMENT

Claims 1-9 are patentable over Fan in view of Abadi.

A. The Claimed Invention

Claim 1 is directed to a network connection control apparatus for granting or rejecting access when a device on a global network demands access to services provided on a local network. The network connection control apparatus comprises authentication means, access permission entry creating means, and control means. The authentication means authenticates the device on the global network in response to a service access request message. The access permission entry creating means creates an access permission entry in response to an access request from the device authenticated by the authentication means, and adds the access permission entry to an access permission list. Upon receiving a data packet sent from the device on the global network, the control means determines whether or not the data packet should be transferred to the local network based on information extracted from the header of the data packet and on the access permission entry contained in the access permission list.

Claims 2-6 depend from claim 1.

Claim 7 is directed to a network connection control method for granting or rejecting access when a device on a global network demands access to services provided on a local network. The method comprises the steps of authenticating the device on the global network in response to a service access request message; creating an access permission entry in response to an access request from the authenticated device and adding the access permission entry to an access permission list; and determining, upon receiving a data packet from a device on the global network, whether or not the data packet should be transferred to the local network based on information extracted from the header of the data packet and on the access permission entry contained in the access permission list.

Claims 8-9 depend from claim 7.

B. Claims 1-9 Are Patentable

In the Final Office Action, claims 1-9 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Fan (U.S. Patent No. 6,219,706) in view of Abadi (U.S. Patent No. 5,315,657). The Examiner has not made an adequate showing to support his rejections.

Fan is directed to an access control for networks. Fan discloses a firewall that may inspect each packet within a data flow to ensure that the packets meet the criteria established by a user's security policy. (See col. 7, lines 35-37). The Examiner agrees that Fan does not disclose or suggest authenticating a device on the global network in response to a service access request message. (See Final Office Action at 3).

Abadi is directed to compound principles in access control lists. In Abadi, the user initially establishes a communications channel over which it wishes to converse with the system resource. (See col. 5, line 34 through col. 6, line 3). This initiates the authentication process

where the user must demonstrate knowledge of a particular private key. (See col. 4, lines 61-64). The entity receiving the authentication request must accurately be able to determine that knowledge of a particular private key implies a particular principle name. (See col. 4, lines 64-67). The user may then make a request of a resource. Access to system resources is control by access control lists associated with each system resource. (See col. 4, lines 7-10). When a user makes a request of a resource, the reference monitor for that resource looks for the requesting user on that resources access control list, and if the user's name is found, the requested access is granted. (See col. 4, lines 10-16). The request for a resource occurs after the communication channel is already established. Thus, the authentication request in Abadi does not include a request for service from the network, as the Examiner claims in the Advisory Action. Because the user in Abadi does not send a service access request message (*i.e.*, a message notifying the gateway of the service requested by the terminal device on the global network), Abadi does not disclose or suggest authenticating the device on the global network in response to a service access request message, as required by the claims.

For the reasons set forth above, neither Fan nor Abadi discloses or suggests authenticating a device on the global network in response to a service access request message, as required by the claims. Thus, it would not have been obvious to one of ordinary skill in the art at the time of the invention to combine the teachings of Fan and Abadi to derive claims 1-9. Accordingly, Appellant respectfully submits that claims 1-9 are allowable over Fan in view of Abadi.

C. Conclusion

Appellant respectfully submits that the subject matter of the claims on appeal is not disclosed or suggested by Fan or Abadi. Thus, the Examiner has not made an adequate showing of obviousness with respect to the subject matter of the rejected claims. Appellant, therefore, respectfully requests reversal of the Examiner's decision to reject claims 1-9 under 35 U.S.C. § 103(a) as being unpatentable over Fan in view of Abadi, and respectfully requests allowance of all pending claims.

Respectfully submitted,

Dated: June 26, 2006

By: 

Marina N. Saito
Registration No. 42,121
SONNENSCHEIN NATH & ROSENTHAL LLP
P.O. Box 061080
Wacker Drive Station, Sears Tower
Chicago, Illinois 60606-1080
(312) 876-8000

VIII. CLAIMS APPENDIX

1. (Previously Presented) A network connection control apparatus for granting or rejecting access when a device on a global network demands access to services provided on a local network, comprising:

authentication means for authenticating the device on said global network in response to a service access request message;

access permission entry creating means for creating an access permission entry in response to an access request from the device authenticated by said authentication means, and adding said access permission entry to an access permission list; and

control means which, upon receiving a data packet sent from the device on said global network, determines whether or not said data packet should be transferred to said local network based on information extracted from the header of said data packet and on the access permission entry contained in said access permission list.

2. (Original) A network connection control apparatus according to Claim 1, wherein said access permission entry creating means extracts access information from an access request packet transmitted from the authenticated device, thereby creating an access permission entry containing a source IP address, a destination IP address, a source port number, a destination port number and a last access permission time.

3. (Original) A network connection control apparatus according to Claim 1, wherein said control means extracts a source IP address, a destination IP address, a source port number and a destination port number from the header of the data packet transmitted from the device on said global network, compares these extracted items of information with the information about

the access permission entry contained in said access permission list, and transfers said data packet to said local network if the two pieces of information correspond in all of the source IP address, destination IP address, source port number and destination port number.

4. (Original) A network connection control apparatus according to Claim 1, wherein said control means eliminates the access permission entry corresponding to a relevant access from said access permission list in accordance with an access termination notification from the device on said global network.

5. (Original) A network connection control apparatus according to Claim 1, wherein said control means calculates the length of time which elapsed from the last access based on a last access permission time stored in the access permission entry which corresponds to the time at which the data packet was received from the device on said global network, and eliminates the access permission entry from said access permission list when the elapsed time exceeds a predetermined reference time.

6. (Original) A network connection control apparatus according to Claim 1, further comprising storage means for storing said access permission list.

7. (Previously Presented) A network connection control method for granting or rejecting access when a device on a global network demands access to services provided on a local network, comprising the steps of:

authenticating the device on said global network in response to a service access request message;

creating an access permission entry in response to an access request from the authenticated device and adding the access permission entry to an access permission list;

determining, upon receiving a data packet from a device on said global network, whether or not said data packet should be transferred to said local network based on information extracted from the header of said data packet and on the access permission entry contained in said access permission list.

8. (Original) A network connection control method according to Claim 7, wherein, in the step of creating the access permission entry, access information is extracted from an access request packet transmitted from the authenticated device, so that an access permission entry can be created which contains a source IP address, a destination IP address, a source port number, a destination port number and a last access permission time.

9. (Original) A network connection control method according to Claim 7, wherein a source IP address, a source port number, a destination IP address and a destination port number are extracted from the header of the data packet transmitted from the device on said global network, and the extracted items of information are compared with information about the access permission entry contained in said access permission list, whereby said data packet is transferred to said local network if the two pieces of information correspond in all of the source IP address, destination IP address, source port number and destination port number.

X. EVIDENCE APPENDIX

Appellant attaches hereto copies of the patents to (1) Fan (U.S. Patent No. 6,219,706), and (2) Abadi (U.S. Patent No. 5,315,657), which were relied upon by the Examiner in his rejection entered on December 27, 2005.

US006219706B1

(12) **United States Patent**
Fan et al.

(10) **Patent No.: US 6,219,706 B1**
(45) **Date of Patent: Apr. 17, 2001**

(54) **ACCESS CONTROL FOR NETWORKS**

(75) Inventors: **Serene Fan**, Palo Alto; **Steve Truong**, Saratoga, both of CA (US)

(73) Assignee: **Cisco Technology, Inc.**, San Jose, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/174,200**

(22) Filed: **Oct. 16, 1998**

(51) **Int. Cl.**[7] **G06F 15/173**
(52) **U.S. Cl.** **709/225**; 709/232; 713/201
(58) **Field of Search** 709/225, 232, 709/229, 220, 217, 250; 713/201

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,606,668		2/1997	Shwed	395/200
5,896,499	*	4/1999	McKelvey	395/187.01
5,898,830	*	4/1999	Wesinger, Jr. et al.	395/187.01
5,951,651	*	9/1999	Lakshman et al.	709/239
6,009,475	*	12/1999	Shrader	709/249
6,052,788	*	8/2000	Wesinger, Jr. et al.	713/201
6,088,796	*	7/2000	Cianfrocca et al.	713/152
6,098,172	*	8/2000	Coss et al.	713/201
6,141,755	*	10/2000	Dowd et al.	713/200

OTHER PUBLICATIONS

Presentation to Customers (described in attached IDS) Beginning Jul. 17, 1997.
Press Release of Cisco Systems, Inc.; www.cisco.com/warp/public/146/1977.html; Oct. 20, 1997.
World Wide Web Page of Check Point Software Technologies, Ltd.; www.checkpoint.com/products/technology/stateful 1.html; downloaded Sep. 18, 1998.

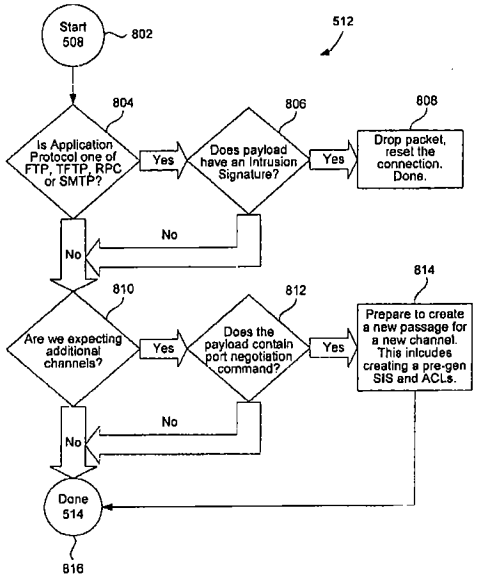
* cited by examiner

Primary Examiner—Zarni Maung
(74) *Attorney, Agent, or Firm*—Beyer Weaver & Thomas, LLP

(57) **ABSTRACT**

An access control system (a firewall) controls traffic to and from a local network. The system is implemented on a dedicated network device such as a router positioned between a local network and an external network, usually the Internet, or between one or more local networks. In this procedure, access control items are dynamically generated and removed based upon the context of an application conversation. Specifically, the system dynamically allocates channels through the firewall based upon its knowledge of the type of applications and protocol (context) employed in the conversation involving a node on the local network. Further, the system may selectively examine packet payloads to determine when new channels are about to be opened. In one example, the firewall employs different rules for handling SMTP (e-mail using a single channel having a well-known port number) sessions, FTP sessions (file transfer using a single control channel having a well known port number and using one or more data channels having arbitrary port numbers), and H.323 (video conferencing using multiple control channels and multiple data channels, which use arbitrary port numbers) sessions.

37 Claims, 11 Drawing Sheets



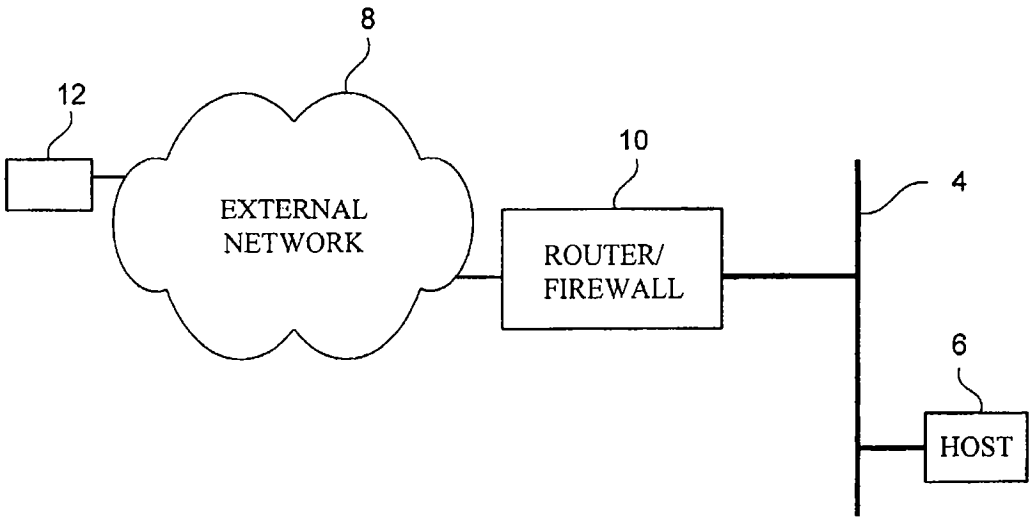


Figure 1

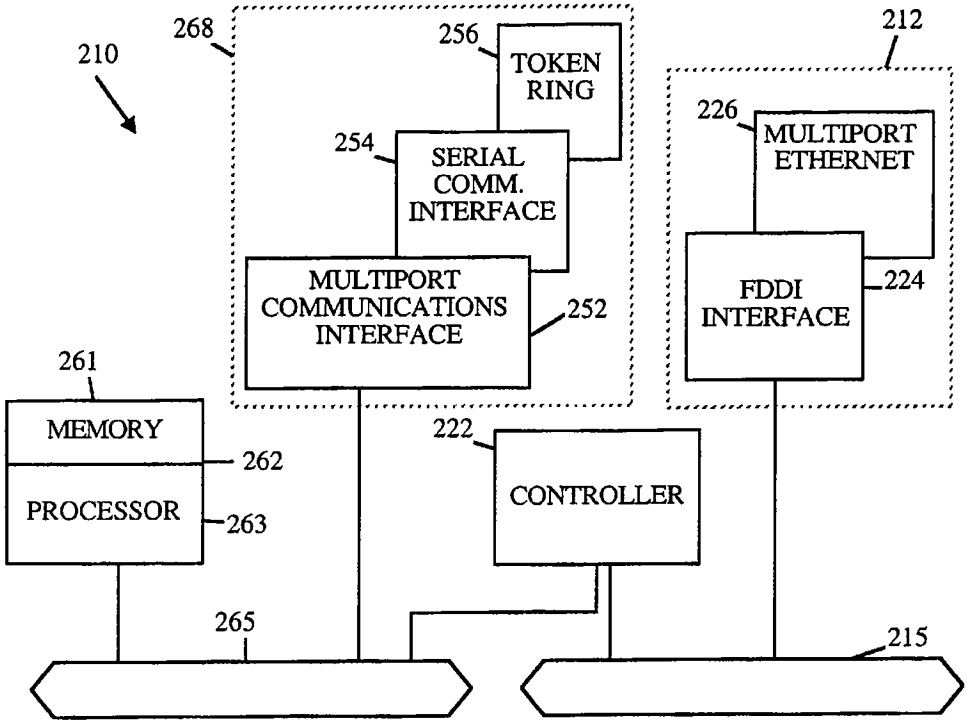

210
268
256
TOKEN RING
254
SERIAL COMM. INTERFACE
MULTIPORT COMMUNICATIONS INTERFACE
252
212
226
MULTIPORT ETHERNET
FDDI INTERFACE
224
261
MEMORY
262
PROCESSOR
263
222
CONTROLLER
265
215

Figure 2

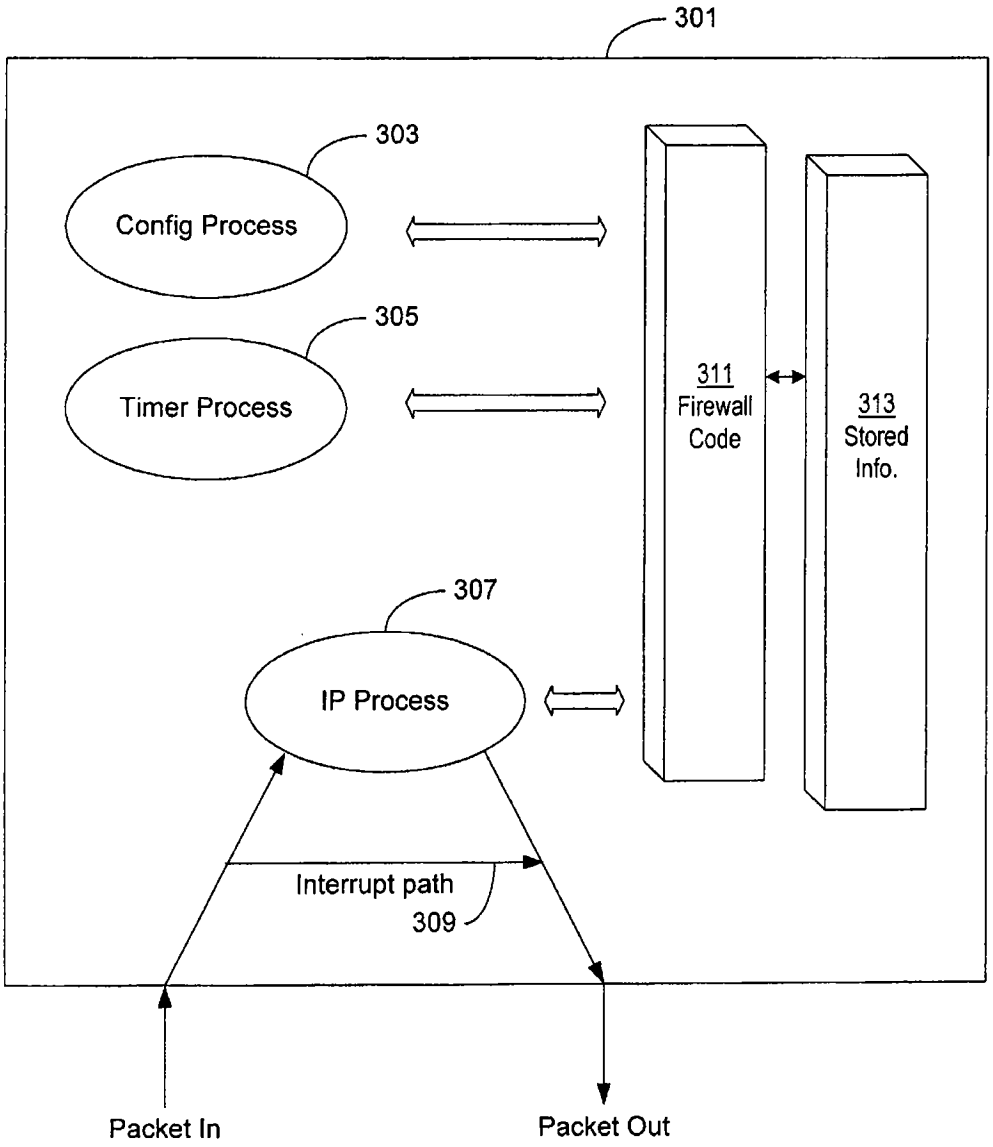

301
303
Config Process
305
Timer Process
311
Firewall
Code
313
Stored
Info.
307
IP Process
Interrupt path
309
Packet In
Packet Out

Figure 3

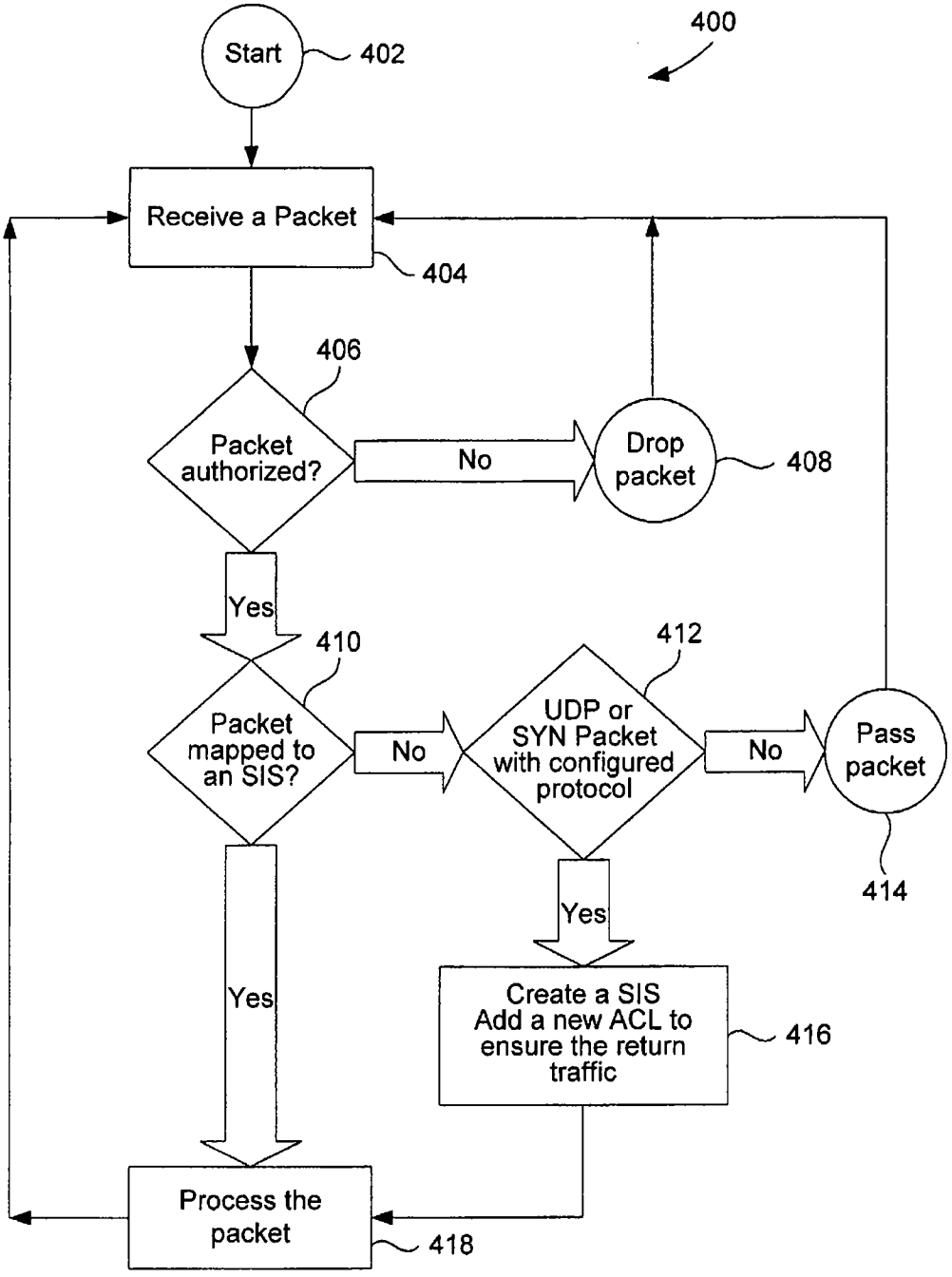

400
Start
402
Receive a Packet
404
406
Packet authorized?
No
Drop packet
408
Yes
410
Packet mapped to an SIS?
No
412
UDP or SYN Packet with configured protocol
No
Pass packet
414
Yes
Yes
Create a SIS Add a new ACL to ensure the return traffic
416
Process the packet
418

Figure 4

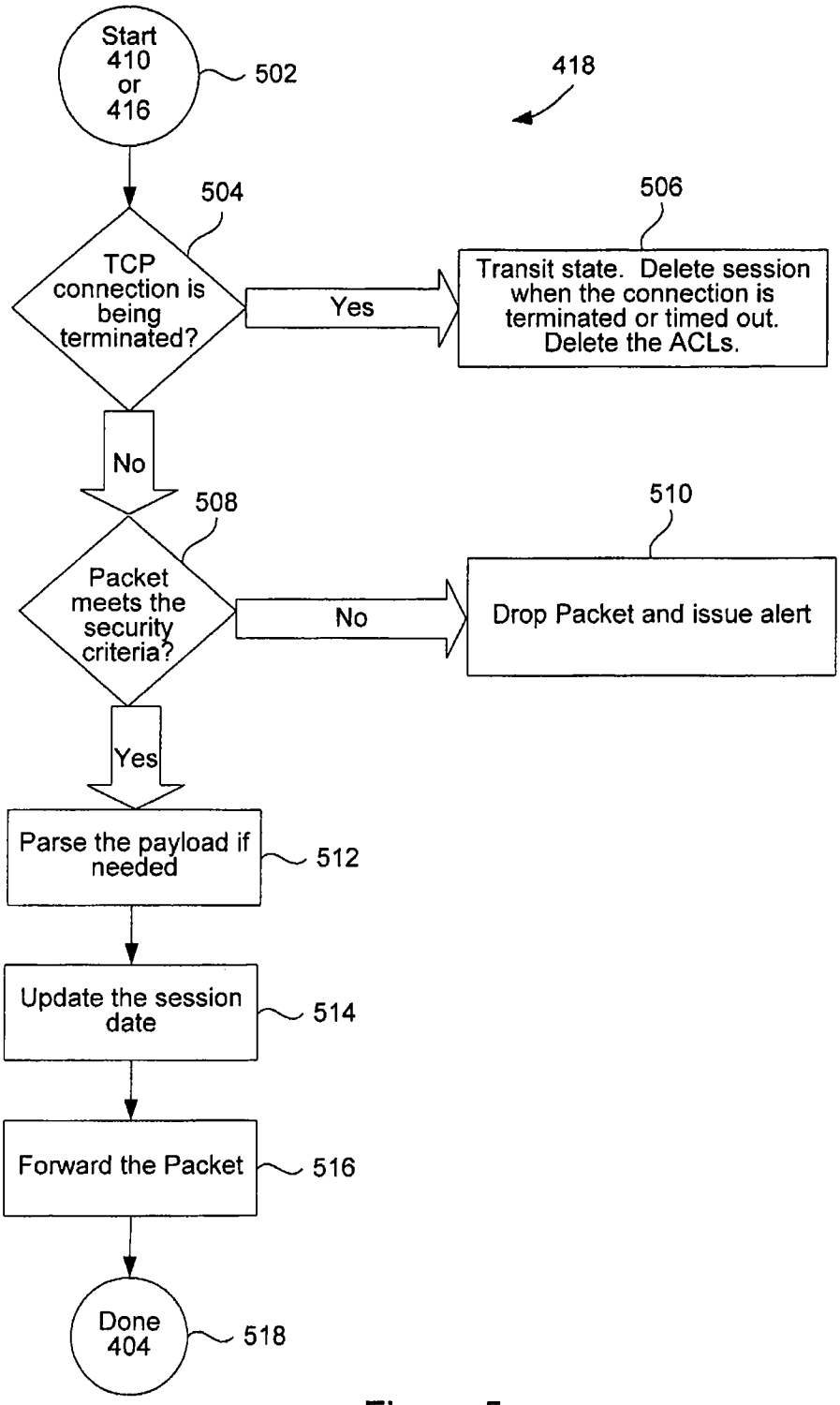

Start
410
or
416
502
418
504
TCP connection is being terminated?
Yes
506
Transit state. Delete session when the connection is terminated or timed out. Delete the ACLs.
No
508
Packet meets the security criteria?
No
510
Drop Packet and issue alert
Yes
Parse the payload if needed
512
Update the session date
514
Forward the Packet
516
Done
404
518

Figure 5

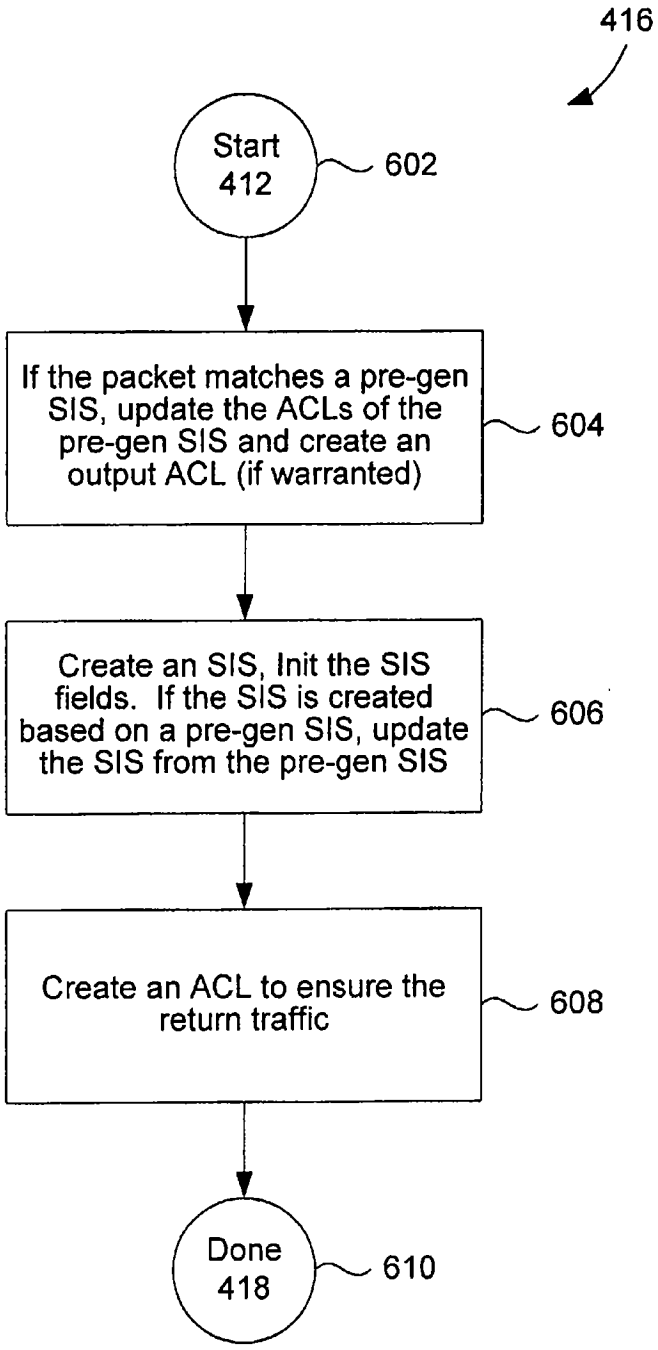

416
Start
412
602
If the packet matches a pre-gen SIS, update the ACLs of the pre-gen SIS and create an output ACL (if warranted)
604
Create an SIS, Init the SIS fields. If the SIS is created based on a pre-gen SIS, update the SIS from the pre-gen SIS
606
Create an ACL to ensure the return traffic
608
Done
418
610

Figure 6

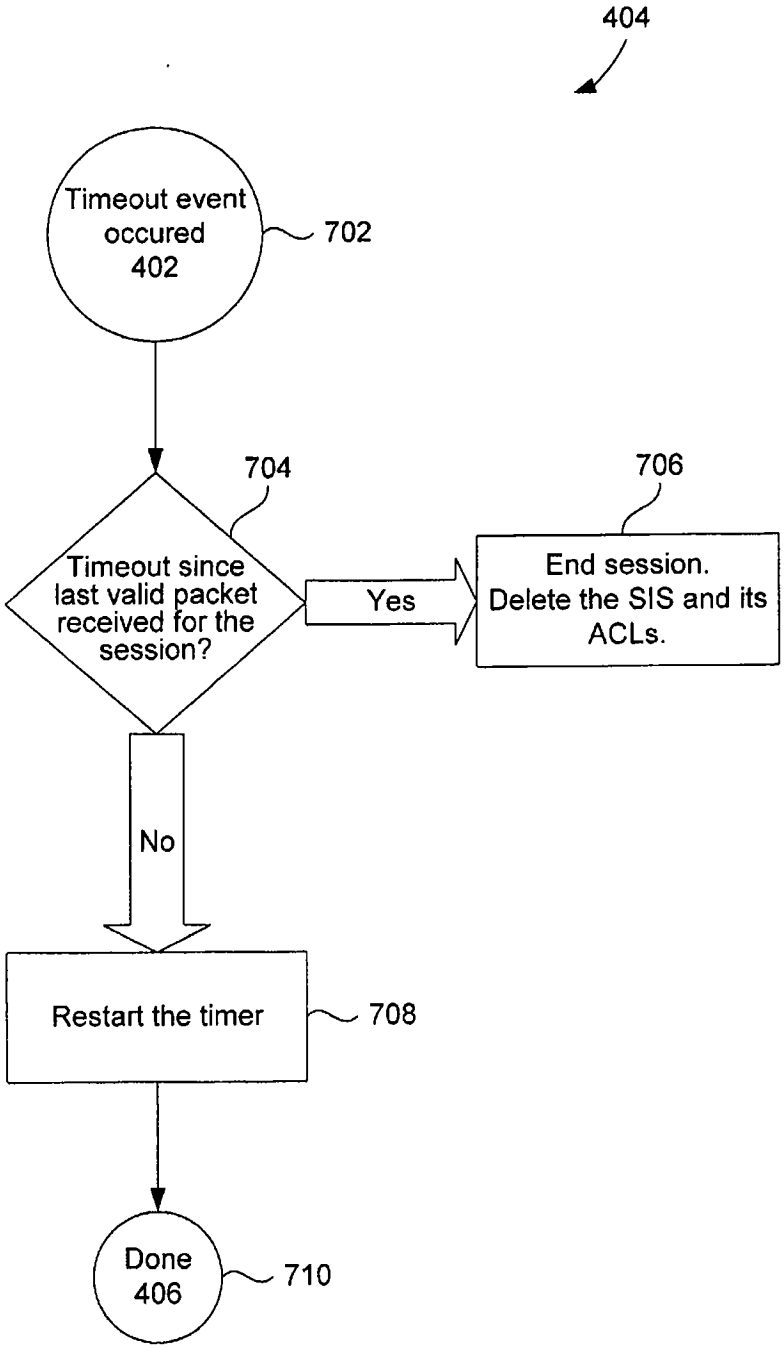

404
Timeout event occured 402
702
704
Timeout since last valid packet received for the session?
Yes
706
End session. Delete the SIS and its ACLs.
No
Restart the timer
708
Done 406
710

Figure 7

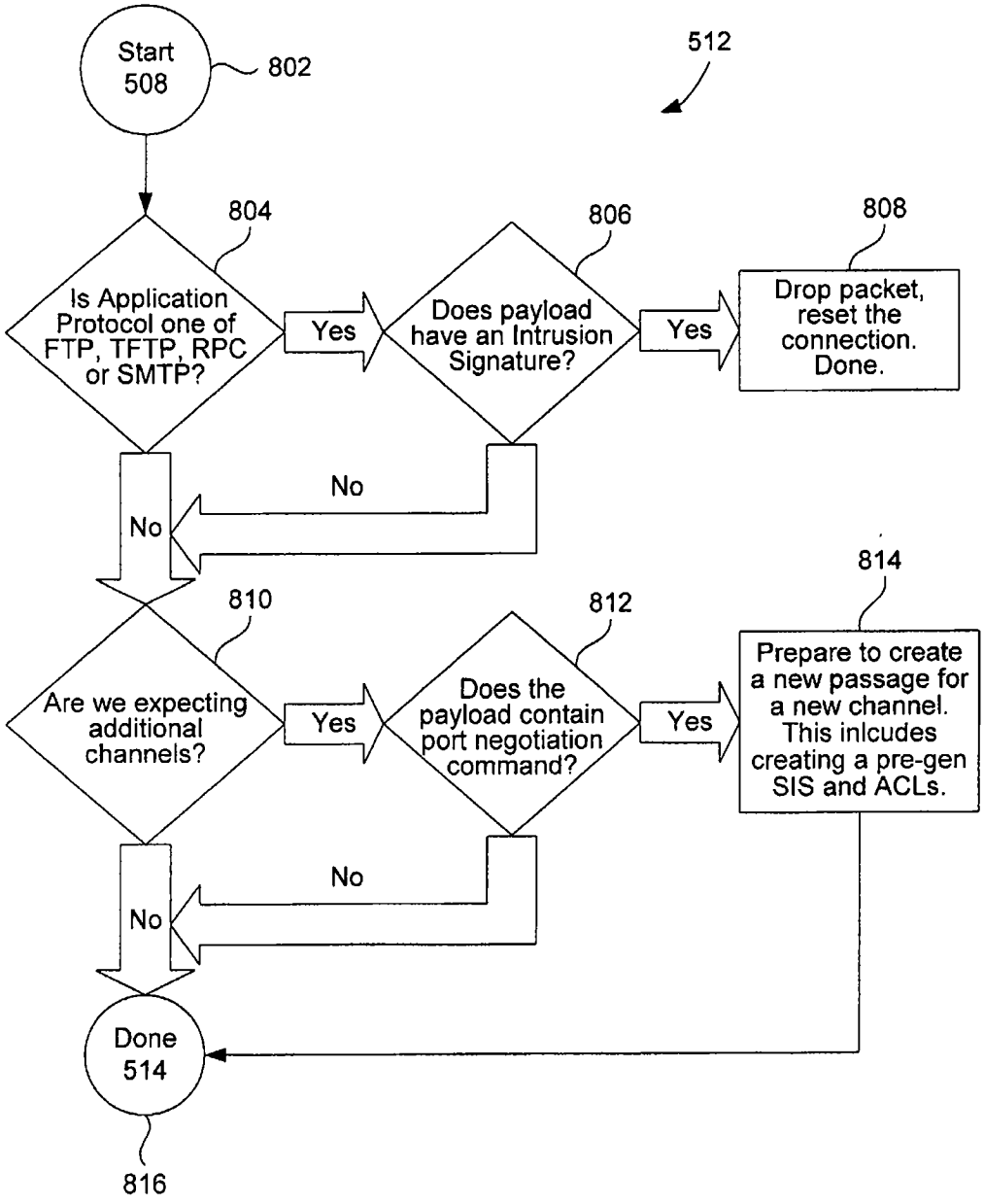

Start
508
802
512
804
Is Application Protocol one of FTP, TFTP, RPC or SMTP?
Yes
806
Does payload have an Intrusion Signature?
Yes
808
Drop packet, reset the connection. Done.
No
No
810
Are we expecting additional channels?
Yes
812
Does the payload contain port negotiation command?
Yes
814
Prepare to create a new passage for a new channel. This inlcudes creating a pre-gen SIS and ACLs.
No
No
Done
514
816

Figure 8

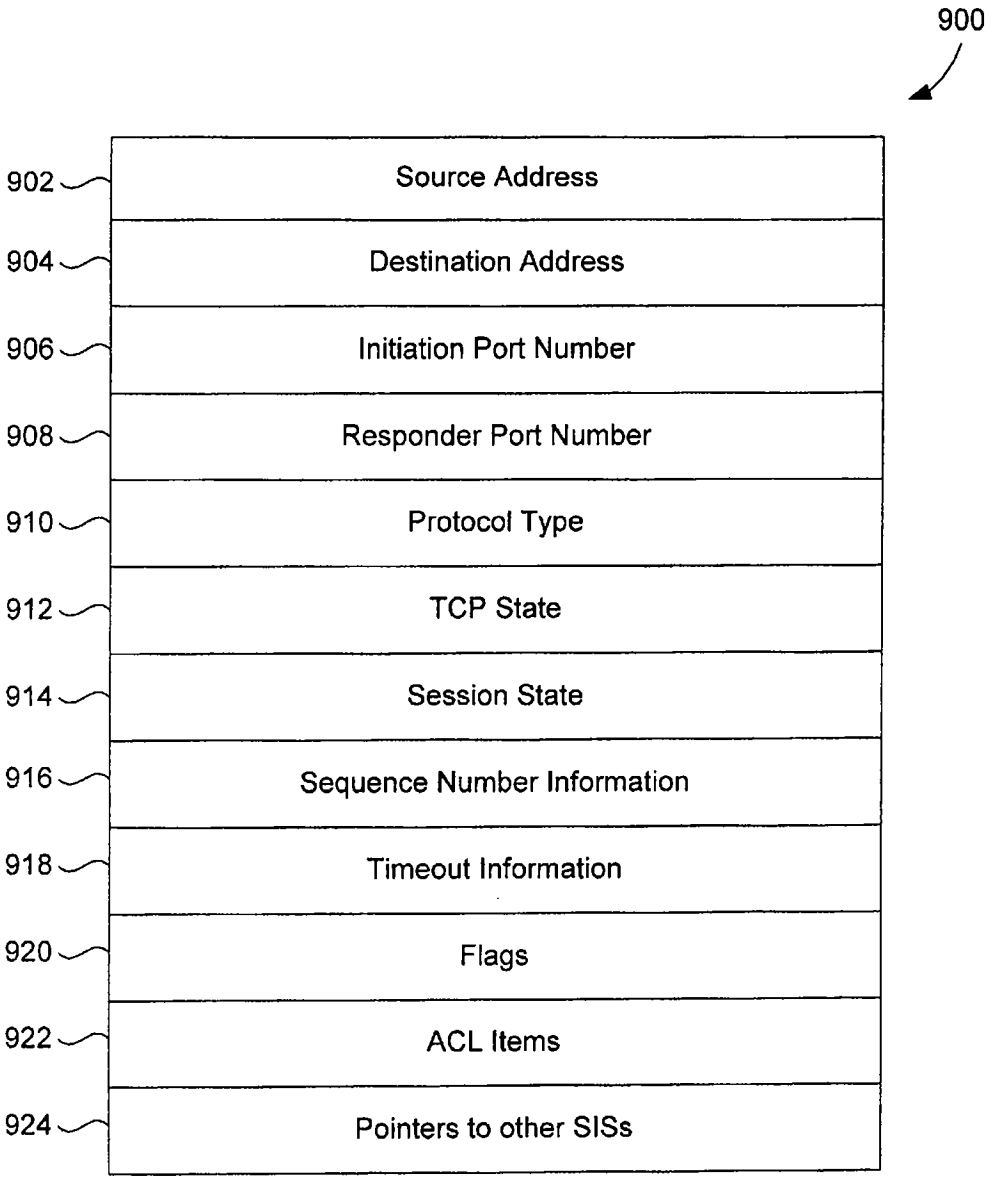

900
902
Source Address
904
Destination Address
906
Initiation Port Number
908
Responder Port Number
910
Protocol Type
912
TCP State
914
Session State
916
Sequence Number Information
918
Timeout Information
920
Flags
922
ACL Items
924
Pointers to other SISs

Figure 9

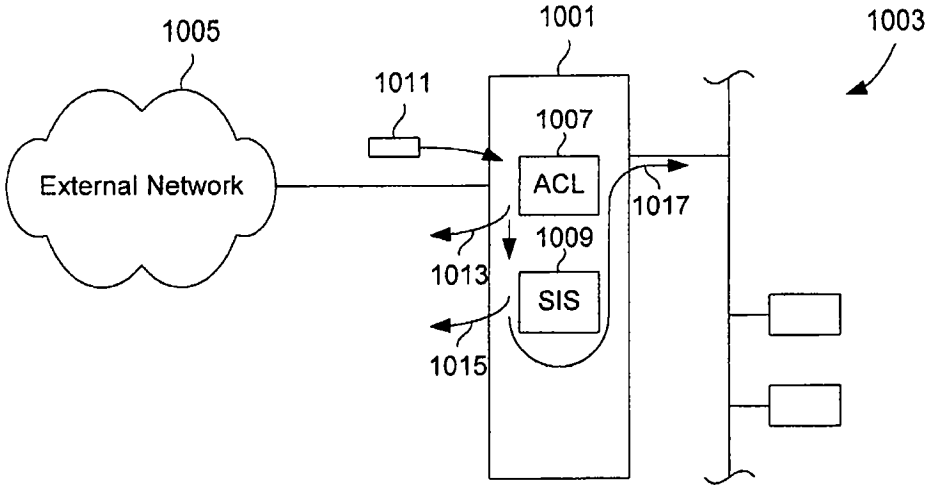


Figure 10A

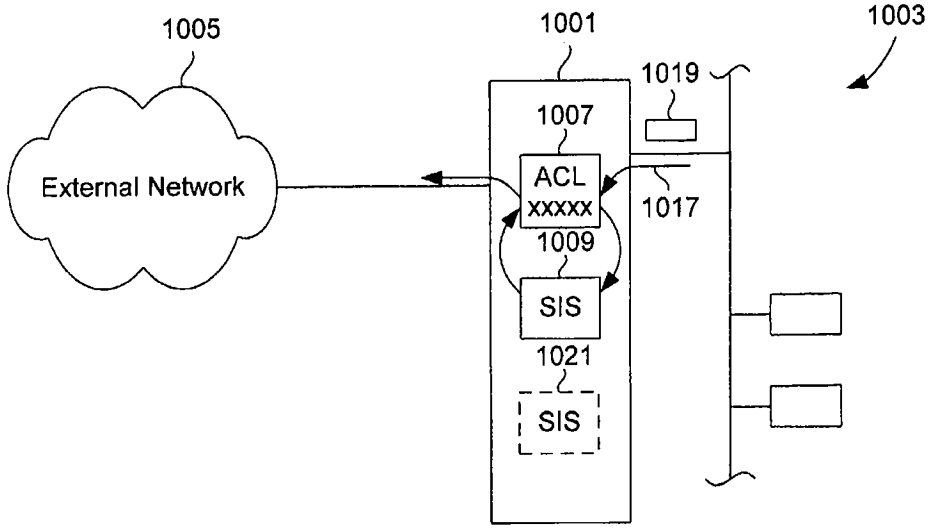


Figure 10B

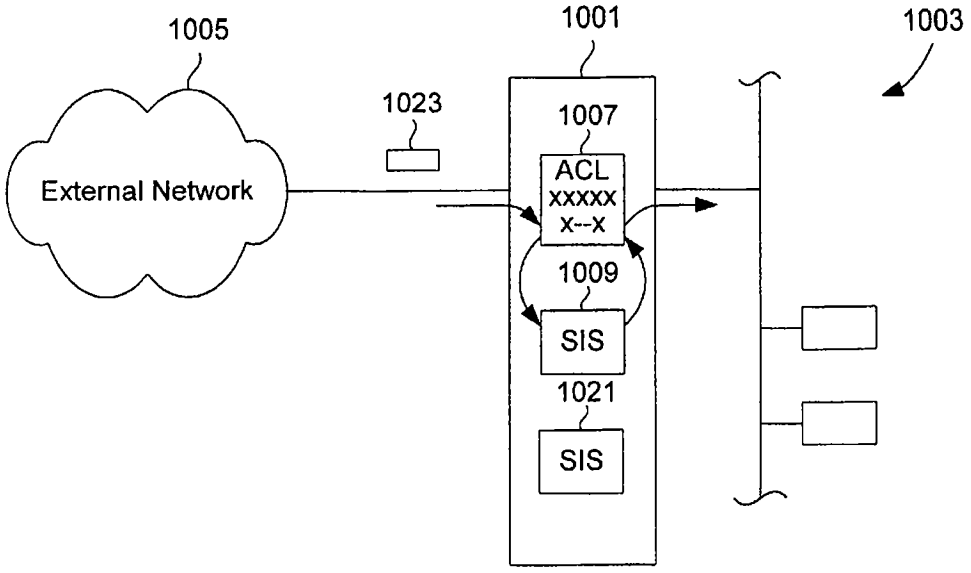

1005
1001
1003
1023
1007
External Network
ACL
XXXXX
X--X
1009
SIS
1021
SIS

Figure 10C

ACCESS CONTROL FOR NETWORKS

BACKGROUND OF THE INVENTION

This invention relates to network firewalls for controlling external access to a particular local network. More particularly, the invention relates to network firewalls having dynamic access control lists.

Firewalls were developed to protect networks from unauthorized accesses. Hackers, corporate spies, political spies, and others may attempt to penetrate a network to obtain sensitive information or disrupt the functioning of the network. To guard against these dangers, firewalls inspect packets and sessions to determine if they should be transmitted or dropped. In effect, firewalls have become a single point of network access where traffic can be analyzed and controlled according to parameters such as application, address, and user, for both incoming traffic from remote users and outgoing traffic to the Internet.

Firewalls most commonly exist at points where private networks meet public ones, such as a corporate Internet access point. However, firewalls can also be appropriate within an organization's network, to protect sensitive resources such as engineering workgroup servers or financial databases from unauthorized users.

Firewalls protect by a variety of mechanisms. Generally, state-of-the art firewall technology is described in "Building Internet Firewalls" by D. Brent Chapman and Elizabeth D. Zwicky, O'Reilly and Associates, Inc. which is incorporated herein by reference for all purposes.

One firewall mechanism involves "packet filtering." A packet filtering firewall employs a list of permissible packet types from external sources. This list typically includes information that may be checked in a packet header. The firewall checks each inbound packet to determine whether it meets any of the listed criteria for an admissible inbound packet. If it does not meet these criteria, the firewall rejects it. A similar mechanism may be provided for outbound packets.

Often, the firewall maintains the access criteria as an access control list or "ACL." This list may contain network and transport layer information such as addresses and ports for acceptable sources and destination pairs. The firewall checks packet headers for source and destination addresses and source and destination ports, if necessary, to determine whether the information conforms with any ACL items. From this, it decides which packets should be forwarded and which should be dropped. For example, one can block all User Datagram Protocol ("UDP") packets from a specific source IP address or address range. Some extended access lists can also examine transport-layer information to determine whether to forward or block packets.

While packet filtering is a very fast firewall technology, it is not, unfortunately, very good at handling protocols that create multiple channels or do not necessarily employ well-known port numbers. A channel is typically defined by a source address, a destination address, a source port number, and a destination port number. In Transport Control Protocol ("TCP"), a channel is referred to as a connection. For some protocols, such as SMTP (electronic mail), only a single well-known destination port is used. Conversations involving these protocols involve only a single channel. For such cases, the packet filtering mechanism will include an ACL item defining allowed accesses using the well-known port number. Because this well-known port number never changes, the ACL item can be set initially and left unchanged during the life of the firewall. Other protocols do not necessarily use well-known port numbers. In these cases, the port number is assigned dynamically. That is, for each new session a different port number may be assigned. Obviously, in these cases, a static packet filtering mechanism must either block all use of this protocol or allow all use, regardless of port number. This represents a significant limitation of standard packet filtering mechanisms.

In addition to single channel protocols, a variety of multi-channel protocols are known and others are being developed. For example, the File Transfer Protocol ("FTP") sets up a control channel using a well-known port and a data channel using a variable port number. The control channel is used to initiate the FTP connection between the clients and a server. Via this control channel, the client and server negotiate a port number for a data channel. Once this data channel is established, the file to be retrieved is transmitted from the server to the client over the data channel. Other newer protocols such as the H.323 protocol used for video conferencing employ multiple control channels and multiple data channels such as channels for transmission of audio information and channels for transmission of video information. The port numbers for these data channels can not be known ahead of time. Static packet filtering mechanisms have difficulty handling FTP and most multi-channel protocols.

Another approach to firewall designs is employed in a "Stateful Inspection" firewall provided by Check Point Software Technology Ltd. In this approach, the firewall inspects not only the packet header but also the packet payload. This allows for the possibility of identifying channels in which the port number or numbers are set by the communicating nodes during a conversation. Specifically, the port numbers of channels about to be opened may be specified in the payload or payloads of packets transmitted over a control channel for a conversation. By inspecting packet payloads in a control channel, the firewall can open a temporary channel corresponding to the port numbers agreed upon by the nodes establishing the session. When the session is terminated, the firewall can reseal the channel associated with those port numbers.

Unfortunately, the firewall implemented by Check Point resides on a PC or a workstation host. Such host must be positioned at the interface of a local network and an external network. Typically, it must be used in conjunction with a router. This configuration limits the flexibility and efficiency of the firewall.

For the above and other reasons, it would be desirable to have an improved firewall design.

SUMMARY OF THE INVENTION

The present invention addresses this need by providing an access control system and method for controlling traffic to and from a local network. The system and procedures of this invention are preferably implemented on a dedicated network device such as a router positioned between a local network and an external network, e.g., the Internet, or between one or more local networks. In this procedure, access control items are dynamically generated and removed based upon the context of an application conversation. Specifically, the procedures of this invention may dynamically allocate channels through the firewall based upon its knowledge of the type of application and protocol (context) employed in the conversation involving a node on the local network. Further, the procedure may selectively examine packet payloads to determine when new channels are about to be opened. In one example, the system employs different

rules for handling SMTP (e-mail using a single channel having a well-known port number) sessions, FTP sessions (file transfer using a single control channel having a well known port number and using one or more data channels having arbitrary port numbers), and H.323 (video conferencing using multiple control channels and multiple data channels, which use arbitrary port numbers) sessions.

One aspect of the invention pertains to methods of limiting access to a local network. The methods may be characterized by the following sequence: (a) receiving a packet; (b) identifying an application associated with the packet; (c) determining whether the packet possesses a predefined source or destination address or port; (d) determining whether the packet meets criteria for a current state of a TCP or UDP session with which it is associated; (e) determining whether to examine the payload of the packet; and (f) examining the packet payload. The method may also include various other operations such as determining whether the packet sequence number falls within a defined sequence window and determining whether the packet has been received after a predetermined timeout period has elapsed.

The process of determining whether the packet meets criteria for a current state may involve determining whether any state transition associated with a TCP or UDP session follows an expected sequence of state transitions (e.g., a TCP FIN packet is received after a session is open). The process of determining whether to examine the payload may involve determining whether the payload may contain an intrusion signature. In a specific embodiment, that involves determining whether the packet is an FTP packet, an RPC, a TFTP packet, or a SMTP packet. If the system identifies an intrusion signature in the packet payload of such packet, it will drop the packet. The process of determining whether to examine the payload may also involve determining whether an additional channel of unknown port number may be opened (e.g., the connection is an FTP control channel or an H.323 channel when less than all data channels have been opened). Assuming that the system determines that an additional channel could be opened, it examines the packet payload to identify a port negotiation command. If such port negotiation command is detected, the system may dynamically modify an access control list to create a path for the additional channel.

The system may also detect when a packet initiates a new session (e.g., it is a TCP SYN packet). When this occurs, the method may involve (i) creating a state entry (e.g., a data structure) for the new session; and (ii) creating one or more access control items allowing passage of packets from a node identified in the packet initiating the new session.

Another aspect of the invention pertains to network devices such as routers which may be characterized by the following features: (a) two or more interfaces configured to connect with distinct networks or network segments; (b) a memory or memories configured to store (i) one or more access control criteria for allowing or disallowing a packet based upon header information and (ii) information specifying the content of an application conversation; and (c) a processor configured to compare packet header information with the access control criteria and determine whether to examine packet payloads based upon the context of the application conversation. The network device may include an operating system which controls the network device to perform functions necessary to control access to the local network and route network traffic. To facilitate rapid processing of packets, the network device may include at least two processors, at least one of which is associated with one of the interfaces.

The memory may be configured to store the access control criteria in the form of an access control list. It may also be configured to store state information such as the state of at least one of a TCP session and a UDP session. It may further be configured with information specifying the context of an application conversation indicating whether a side channel may be opened for the application.

The processor may be configured to examine packet payloads when context information in the memory indicates that a side channel may be opened. In such cases, the processor may initiate steps to dynamically modify the access control criteria when a new side channel opens.

These and other features and advantages of the present invention will be presented in more detail below with reference to the associated drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a diagram illustrating how a firewall of this invention may be integrated in a network.

FIG. **2** is a block diagram of a router that may be used in this invention.

FIG. **3** is a block diagram of a computer architecture that may be employed with this invention.

FIGS. **4–8** are flow charts depicting a preferred method by which the firewalls of this invention may protect a local network.

FIG. **9** is diagram of a State Information Structure (a data structure) used in a preferred implementation of this invention.

FIGS. **10A–10C** depict an FTP session using a firewall/router in accordance with an embodiment of this invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

1. System Structure and Architecture

FIG. **1** illustrates a general arrangement by which a local network allows its hosts (e.g., a host **6**) to communicate with external nodes located on an external network **8** such as the Internet. Typically local network **4** is connected to external network **8** via a router **10** which routes packets between external network **8** and local network **4**.

In this invention, router **10** may also double as a firewall that protects local network **4** from potentially dangerous accesses from external network **8**. When acting as a firewall, a router **10** will, under certain circumstances, allow host **6** to initiate a conversation with an external node **12** that is connected to external network **8**. If router/firewall **10** allows host **6** to initiate such a conversation, it must also allow appropriate return communications from node **12** to host **6**. Details of how router/firewall **10** allows such conversations and yet protects the local network will be detailed below, in one embodiment.

Generally, a firewall of this invention may be specially constructed for the required purposes, or it may be a general-purpose programmable machine selectively activated or reconfigured by a computer program stored in memory. The processes presented herein are not inherently related to any particular router or other network apparatus. Preferably, the invention is implemented on a network device designed to handle network traffic. Such network devices typically have multiple network interfaces including frame relay and ISDN interfaces, for example. Specific examples of such network devices include routers and switches. For example, the firewalls of this invention may be

specially configured routers such as specially configured router models 1600, 2500, 2600, 3600, 4500, 4700, 7200, and 7500 available from Cisco Systems, Inc. of San Jose, Calif. A general architecture for some of these machines will appear from the description given below. In an alternative embodiment, the firewall may be implemented on a general-purpose network host machine such as a personal computer or workstation. Further, the invention may be at least partially implemented on a card (e.g., an interface card) for a network device or a general-purpose computing device.

Referring now to FIG. **2**, a router **210** suitable for implementing the present invention includes a master central processing unit (CPU) **262**, low and medium speed interfaces **268**, and high-speed interfaces **212**. When acting under the control of appropriate software or firmware, the CPU **262** is responsible for such router tasks as routing table computations and network management. It is also responsible for creating and updating an Access Control List, comparing incoming packets with the current Access Control List, generating State Information Structures, inspecting packet headers and payloads as necessary, enforcing the state of a session, etc. It preferably accomplishes all these functions under the control of software including an operating system (e.g., the Internet Operating System (IOS®) of Cisco Systems, Inc.) and any appropriate applications software. CPU **262** may include one or more microprocessor chips **263** such as the Motorola MPC860 microprocessor, the Motorola 68030 microprocessor, or other available chips. In a preferred embodiment, a memory **261** (such as non-volatile RAM and/or ROM) also forms part of CPU **262**. However, there are many different ways in which memory could be coupled to the system.

The interfaces **212** and **268** are typically provided as interface cards (sometimes referred to as "line cards"). Generally, they control the sending and receipt of data packets over the network and sometimes support other peripherals used with the router **210**. The low and medium speed interfaces **268** include a multiport communications interface **252**, a serial communications interface **254**, and a token ring interface **256**. The high-speed interfaces **212** include an FDDI interface **224** and a multiport ethernet interface **226**. Preferably, each of these interfaces (low/medium and high-speed) includes (1) a plurality of ports appropriate for communication with the appropriate media, and (2) an independent processor such as the 2901 bit slice processor (available from Advanced Micro Devices corporation of Santa Clara Calif.), and in some instances (3) volatile RAM. The independent processors control such communications intensive tasks as packet switching, media control and management. By providing separate processors for the communications intensive tasks, this architecture permits the master microprocessor **262** to efficiently perform routing computations, network diagnostics, security functions, etc.

The low and medium speed interfaces are coupled to the master CPU **262** through a data, control, and address bus **265**. High-speed interfaces **212** are connected to the bus **265** through a fast data, control, and address bus **215** which is in turn connected to a bus controller **222**. The bus controller functions are provided by a processor such as a 2901 bit slice processor.

Although the system shown in FIG. **2** is a preferred router of the present invention, it is by no means the only router architecture on which the present invention can be implemented. For example, an architecture having a single processor that handles communications as well as routing computations, etc. would also be acceptable. Further, other types of interfaces and media could also be used with the router.

Regardless of network device's configuration, it may employ one or more memories or memory modules (including memory **261**) configured to store program instructions for the network operations and access control functions described herein. The program instructions may specify an operating system and one or more applications, for example. Such memory or memories may also be configured to store access control criteria (e.g., an ACL), state information (specifying the context of a network session for example), etc.

Because such information and program instructions may be employed to implement the access control systems/methods described herein, the present invention relates to machine readable media that include program instructions, state information, etc. for performing various operations described herein. Examples of machine-readable media include, but are not limited to, magnetic media such as hard disks, floppy disks, and magnetic tape; optical media such as CD-ROM disks; magneto-optical media such as floptical disks; and hardware devices that are specially configured to store and perform program instructions, such as read-only memory devices (ROM) and random access memory (RAM). Examples of program instructions include both machine code, such as produced by a compiler, and files containing higher level code that may be executed by the computer using an interpreter.

FIG. **3** is a system diagram of router or other network device **301** that may implement a firewall in accordance with this invention. As shown network device **301** includes various processes and paths that form part of an operating system for the network device. These may include configuration processes **303**, timer processes **305**, IP processes **307**, and interrupt paths **309**. IP processes **307** and interrupts **309** are provided for routine packet handling functions as illustrated in the figure. In addition to these processes and paths, network device **301** includes firewall code **311** for executing firewall functions in response to requests from processes **303**, **305**, and **307** and interrupts **309**. In a preferred embodiment, firewall code **311** may include both an engine that handles transport layer functions and various inspection modules, each of which is dedicated to handling a specific application protocol (e.g., FTP, H.323, etc.). In a further preferred embodiment, firewall code **311** is integrated with the remainder of the network device's operating system.

Firewall code **311** may make use of various lists, data structures, and other stored information (collectively indicated by reference numeral **313** in FIG. **3**). Examples include access control lists, state information structures (described below), timers, and various lists.

Regarding the operating system, it may require execution of code **311** under various circumstances associated with packet processing. In one example, configuration processes **303** specify that the FTP protocol is to be inspected. Thus processes **303** may ask code **311** to configure an access control list to allow initiation of an FTP session. Timer processes **305** may indicate to code **311** that a particular session has timed out. In this case, the firewall code **311** may delete any state information structure for that session as well as the associated ACL items. Still further IP processes **307** and interrupts **309** may call firewall code **311** during the course of processing a packet to determine whether it meets certain ACL items or to determine whether its payload should be inspected.

2. Firewall Process

Overview

Network communications at high levels, such as at the application layer, may be referred to as "conversations." An

"application conversation" may have one or many "channels" (also referred to as "sessions" or "socket pairs"). These terms were chosen to cover at least TCP and UDP communications. In TCP, each channel represents a separate "connection." In UDP, which is connectionless, each channel is defined by a unique combination of source and destination IP addresses and port numbers. All UDP packets received within a defined timeout period and having the same unique combination of addresses and port numbers are deemed to belong to the same session or channel.

An application conversation may include only a single well-known channel as in the case of SMTP, HTTP, and Telnet or it may contain many channels as in the case of certain multimedia applications (e.g., H.323 and RealAudio). Still other application conversations may have variable numbers of channels as in the case of FTP and TFTP which create a new data channel each time a different file is transferred from server to client. The present invention handles all of these situations.

Like packet filtering, the access control of this invention examines network and transport-layer information. In addition, it examines application layer protocol information (such as FTP) to learn about and inspect the state of TCP or UDP sessions. This mechanism dynamically creates and deletes temporary openings in the firewall by temporarily modifying access lists to change packet filtering criteria. Preferably, the dynamically created access control list items are stored in memory in the network device's network interface. A firewall of this invention may also maintain state information in its own data structures (referred to herein as State Information Structures or "SISs") and use that information to create the temporary entries (by dynamically modifying its ACL, for example). Thus, a firewall may retain state information that is not retained in the access list entries. A firewall may inspect each packet within a data flow to ensure that the state of the session and packets themselves meet the criteria established by a user's security policy. State information is used to make intelligent permit/deny decisions. When a session closes, its temporary ACL entry is deleted, and the opening in the firewall is closed.

A firewall may monitor each application on a per-connection basis for comprehensive traffic control capability. The firewall watches application sessions, notes the ports each session is using and opens the appropriate channels for the duration of the session, closing them when the session is finished. Specifically, when a newly authorized session is registered, the system may create a new SIS and any new ACL items for the session. Thereafter, packets transmitted to and from the hosts involved in the connection are allowed to pass back and forth across the firewall so as long as the ACL items allow a transmission.

The firewalls of this invention preferably consider the TCP or UDP session state. In fact, a firewall may base decisions on the state of its sessions. To do so, it may maintain a record of the state of each connection going through. Also, the firewalls preferably keep track of items such as: how long was the last transmitted packet in this session, are the sequence/acknowledgment numbers climbing as expected, was the session initiated from the inside or outside, is the session still open or has it been closed, and what port or ports are the return data channels using?

The firewalls of this invention may enable a firewall to support protocols that involve multiple data channels created as a result of negotiations in the control channel. As mentioned, many Internet and multimedia applications that use a "well-known" port address to open an initial control connection often use different, dynamically chosen ports for data traffic. It is impossible to predict which ports these applications may use in a given connection, and some of them may use multiple channels over several ports. Thus, depending upon the type of application conversation, the firewall may also monitor the payloads of the packets it allows to pass. This may be the case when, for example, a control channel connection is in a state in which ports for additional channels are negotiated over a control channel. This allows the firewall to determine which additional channels should be dynamically opened to the firewall.

Security Access Policy

Initially, an administrator or other authorized person may create a "security access policy" for the firewall. The purpose of this policy is to generally define how to protect the local network. Often the policy will protect the network from all uninvited sessions initiated externally. In such cases, the policy may specify which local nodes may participate in conversations outside the local network and the protocols under which those conversations may take place. In addition, the security access policy may specify particular times when access will be permitted to particular users operating under particular protocols. For example, it may be desirable to provide a security access policy in which certain users cannot communicate outside the local network during nonbusiness hours.

A security access policy typically will specify that few if any uninvited packets from outside the local network are permitted to enter. Then when a local node initiates a conversation with an external node, the firewall must anticipate that packets in response will be addressed to the local node. It does this by adjusting its ACL to include items allowing passage of certain packets having the external node's address.

While a typical security access policy will allow no uninvited packets from external sources, some policies may allow some limited conversations initiated by external nodes. Such policies may place restrictions on the protocols that could be used by any external nodes initiating a conversation.

Various combinations of matching (or not matching) packet header fields can be used to support a policy. Examples of specific fields that may be examined include IP destination address, IP source address, IP protocol field, TCP source port, TCP destination port, TCP flags field, SYN alone for a request to open a connection, SYN/ACK for a connection confirmation, ACK for a session in progress, and FIN for session termination. All or some of that information may be compared against an ACL and/or used by the firewall engine to determine whether the packet is appropriate given the current state of the session.

In a specific example, an access control list item may specify the addresses of the communicating hosts (or the sub-networks of one or both of these hosts) and the protocol under which they communicate (identified by a port number for example). More specifically, for example, if the security access policy prevents SMTP sessions initiated from IP host 1.1.1.1. with a destination address 2.2.2.2. then the packet filter would discard packets that have IP destination address=2.2.2.2., IP source address=1.1.1.1., IP protocol=6 (for TCP), and Destination port=25 (for SMTP). Such criteria may represent static Access Control List items.

The access policy may also restrict a given interface on the router or other network device implementing the firewall of this invention. The interface may specify a particular type of media such as FDDI, Ethernet, Token Ring, etc. Other fields may be considered; the policy may add a check "ACK bit not set" to guard against the connection being a non-SMTP connection initiated outgoing from port **25**, for example.

Similarly, packet filters may be employed for other protocols such as Novell IPX and Apple Talk protocols, because their formats are well documented and understood.

Process Flow Details

One implementation of the present invention is detailed in the process flow charts depicted in FIGS. **4–8**. FIG. **4** presents a high level overview of the process. This particular implementation of the process is referred to by a reference number **400**. The process begins at **402** and receives a new packet at the firewall at a step **404**. Note that a packet may be associated with a session that has "timed out." Timeout criteria are further detailed in a process flow diagram presented in FIG. **7**. Briefly, if there is too much time between receipt of consecutive packets belonging to the same session, the firewall will not allow the subsequent packets of the session to pass.

Assuming that the current packet meets the timeout criteria, the firewall next determines whether it meets additional authorization criteria at a decision step **406**. These criteria may include such items as the time of day when packets can be sent, source and destination IP addresses, protocols to which the packets may belong (identified by port numbers), and combinations thereof. Such authorization criteria may take the form of ACL items. Most, if not all, of this information may be obtained by examining the packet header in the manner of a traditional packet filter. The ACL items may be divided into static items and dynamic items. Static items usually result directly from the user's security access policy (as described above). Dynamic items may be generated on the fly, typically to allow return traffic for sessions or applications initiated with nodes on the local network.

If the firewall determines at decision step **406** that the packet is not authorized, it drops the packet at step **408**. Optionally, the packet is logged at this point. If, on the other hand, the firewall determines that the packet is authorized at decision step **406**, it next determines whether the packet is mapped to a currently existing state information structure ("SIS"). As detailed below, these are data structures that maintain "state" information about a currently existing session.

If the firewall determines that no corresponding SIS exists for the current packet, it next determines whether the packet is a UDP or a TCP SYN packet of a configured protocol at a decision step **412**. For TCP protocols, a request for a new connection is made with a SYN packet. If the firewall determines that the current packet is not a UDP packet or a TCP SYN packet of a configured protocol (i.e., decision step **412** is answered in the negative), it simply passes that packet on to the destination. If, on the other hand, the firewall determines at decision step **412** that the current packet is in fact a UDP or TCP SYN packet of the appropriate protocol, it realizes that a new connection is being opened and should be watched. Therefore, it creates a new SIS at a step **416**. It concurrently adds any necessary ACL items to ensure that return traffic (from the destination) can pass through the firewall, assuming that such return traffic meets other security criteria. The steps of creating a new SIS and associated ACL items will be further detailed in a process flow chart depicted in FIG. **6**.

Note that not all protocols are necessarily monitored as sessions (configured protocols). The network administrator may decide that some protocols (e.g., HTTP) need not be monitored. For such protocols, the firewall understands that no SIS need be created when it encounters a packet of such protocol. It simply passes the packet as indicated in step **414**. This does not necessarily create a security issue as the packet must still be authorized at step **406**.

After the firewall creates the new SIS and associated ACL items at step **416**, it further processes the packet at a step **418**. Similarly, if the firewall determines that the packet it receives is currently mapped to an SIS at decision step **410** (i.e., decision step **410** is answered in the affirmative), it processes the packet per step **418**. Step **418** is further detailed in FIG. **5**. After the packet has been processed according to this procedure, the firewall directs process control back to step **404** where it awaits the next packet.

Turning now to FIG. **5**, the process associated with step **418** is detailed. This process begins at **502** (corresponding to either step **410** or step **416** in process **400**) and follows with a decision step **504** in which the firewall determines whether a TCP connection is being terminated. This is determined by simply identifying an appropriate termination flag in the packet header. These flags may be either the finish (FIN) flag or the reset (RST) flag. Assuming that the firewall determines that the TCP connection is being terminated, it then transitions to a closing or closed state at a step **506**. At the appropriate time or upon receipt of a final packet for the connection, the SIS for that connection is terminated and the associated ACL items for that connection are also deleted. At this point, the system may also provide an audit trail which details connections by recording time stamps, source hosts, destination hosts, ports, total number of bytes transmitted, etc.

If the firewall determines at decision step **504** that the TCP connection is not to be terminated, it next determines whether the packet meets certain security criteria at a step **508**. It may do this by examining the packet header. These security criteria are typically associated with the particular session to which a packet may belong. Examples include ensuring that the packet sequence number falls within a defined range of sequences (a "sequence window"), the packet type is as expected for a given session state, and the packet header meets ACL items associated with the particular session. The state of a given session may be enforced by ensuring that state transition packets arrive in the expected order (e.g., a SYN packet is not received while a TCP session is in an "open" state.) If the firewall finds that the current packet does not meet the criteria specified at step **508**, it drops the packet and optionally issues an alert at a step **510**.

Assuming that the packet meets the security criteria, as determined at step **508**, the firewall next parses the packet payload if necessary as indicated at process step **512**. The parsing procedure is further detailed in a flow chart depicted in FIG. **8**. After the payload parsing is completed, if needed, the firewall may next update the current session state at a step **514**. It may do this if the current packet indicates a state transition.

In one specific embodiment, there are four states for a TCP connection: closed, opening, open, and closing. The transition between closed and opening may occur when a SYN packet is received for a new session. The transition between opening and open states may occur when a SYN/ACK packet is received. The transition to a closing state may occur when a FIN packet is received. Finally, the transition to the closed state may occur upon receipt of a reset packet. In a specific embodiment, UDP communications include the following states: opening, open, and closed. The transition to opening occurs when a first UDP packet for a new session is received. That is, when a UDP packet is received for which there is no existing SIS. The transition from opening to open occurs when the first reply packet to an initial UDP packet is received. The transition to closed occurs when a UDP session times out.

Note that in step **508**, the system may drop the packet if it does not meet expected state criteria. Such criteria may require that state transitions follow an expected sequence: e.g., closed, opening, open, and closing for TCP sessions. The system might then drop FIN packets received while a TCP session is in the closed state or it might drop SYN packets while a TCP session is in the open or closing state.

Step **514** may also involve updating the sequence window for a current session based upon the sequence number of the current packet. The size of the sequence window may be dictated by the network traffic. In a specific embodiment, the sequence window is set at about 1 to 2000 for congested networks and at about 7000 to 9000 for uncongested networks. Preferably, the firewall tracks the sequence number of the packets it receives and the acknowledged sequence number of TCP connections. When the sequence number of a transmitted packet is not in an expected window range as defined based upon the acknowledged sequence window, the packet will be dropped (step **510**). The sequence number of the most recent ACK packet may be maintained in the SIS to define the sequence window of allowable packet sequence numbers. Other bookkeeping tasks may be performed at step **514** and updated in the appropriate fields of the SIS.

Finally, after any necessary updates are performed at step **514**, the firewall forwards the packet to its destination at a process step **516** and the process is completed at **518** (corresponding to step **404** in process **400**).

The process associated with creating a new SIS and adding any new ACL items to ensure return traffic (step **416** of process **400**) is depicted in FIG. **6**. The process begins at **602** (corresponding to decision step **412** of process **400**). Then, if the current packet matches a "pre-gen SIS" (described below), the system updates the ACL items of the pre-gen SIS and may create one or more output ACL items (if warranted). This is accomplished at a process step **604**.

A pre-gen SIS is created when the firewall determines that a side channel or data channel is about to be opened. As explained below, this determination may be made when packet payloads of certain protocols are examined. When the firewall finds a payload marker suggesting that a side/data channel is about to be opened, it prepares for the new connection (associated with the new channel) by creating a precursor (pre-gen) SIS. At this point, the firewall may only know the destination port (as indicated by a port negotiation command in the payload). The ACL items created for the pre-gen SIS may specify this destination port, but they cannot specify the source port, as this is as yet unknown. The second port can be specified when the firewall receives the SYN packet for the new side channel. Such SYN packet will specify the source port and this information may now be added to ACL associated with the pre-gen SIS at step **604**. If the packet under consideration does not match a pre-gen SIS, step **604** is skipped.

Next, at a step **606**, the firewall creates a fresh SIS (for a current UDP or TCP SYN packet) and initializes its fields. If the new SIS is created based upon a pre-gen SIS, some of the initial information is taken from the pre-gen SIS. Finally, at a step **608**, the firewall creates one or more ACL items to ensure return traffic is permitted for the new session. Note that if a pre-gen SIS existed, the ACLs may have been created at step **604**. Without these new ACL items, it is likely that return traffic (presumably from the external network) would be blocked. The new ACL items will typically allow packets from the external node IP address to a local node IP address (as identified in the initial TCP SYN or UDP packet for the SIS) and the associated port number. The process is concluded at **610** (corresponding to step **418** of process **400**).

The timeout provisions are detailed in FIG. **7**. These provisions were discussed above with reference to step **404** of process **400**. Preferably, the timeout provisions are implemented via interrupts and can be triggered at any stage in the firewall process. These provisions are designed to end a session if there is too great a delay between successive packets in that session. Leaving a firewall passage open for too long a time exposes the local network to a potential security problem.

The process begins at **702** (corresponding to step **402** of process **400**) and includes a decision step **704** in which the firewall determines whether there has been a timeout since the last valid packet was received for the session represented by an SIS. In a specific embodiment, the timeout is 30 seconds between successive UDP packets and 3600 seconds between successive TCP packets. Preferably, these timeout periods are configurable. If the firewall determines that the timeout period has been exceeded (by receipt of an interrupt for example), it ends the session and deletes the associated SIS and ACL items at a process step **706**. If, on the other hand, the timeout period has not been exceeded (decision step **704** is answered in the negative), the firewall restarts the appropriate timer when the next packet for that session is received. See process step **708**. The process is then completed at **710** which corresponds to step **406** in process **400**.

To provide a flexible but secure firewall, a security algorithm must examine packet payloads. Preferably, the payload is examined under only certain conditions. By preventing the payload from being examined in all cases, the performance of the system is improved. In a preferred embodiment, the payload is only examined under two circumstances. First, the payload may be examined to identify any intrusion signatures. Certain types of intrusion attempts may be detected by comparing payloads with well-known intrusion signatures. In a specific embodiment all packets of FTP, RPC, TFTP, and SMTP are examined for intrusion signatures. Packet payloads of other protocols are not examined for such signatures in this specific embodiment.

Second, the payload is examined when there is a possibility that an additional channel may be opened. When this is a possibility, the firewall of this invention watches packet payloads to determine whether a port negotiation command has been detected. As noted, some application conversations involve multiple channels. Often there are one or more control channels and one or more data channels. H.323 video conferencing, for example, includes up to three control channels and four data channels. One data channel involves transmission of audio data from a first party, another data channel involves transmission of video data from the first party, another data channel involves transmission of audio data from a second party, and the final data channel involves transmission of video data from the second party. Each new data channel includes a port number that can not be known ahead of time. Each new channel requires a dynamic adjustment of the firewall to temporarily allow data to pass via that channel. The generation of a new channel is prefaced by a "port negotiation command" in a control channel.

In a preferred embodiment, only the payloads of control channels are examined for port negotiation commands. This is because data channel payloads do not indicate that additional channels may be opened. Further control channel payloads are examined only when there is a possibility that an additional channel may be opened. In a H.323 conversation, for example, when all seven channels have been opened, there is no need to further monitor the payloads of the control channels. And, in a NetMeeting

videoconference, the system preferably inspects the TCP control channel used to establish media channels. This control channel contains information that opens new media channels. The system watches it to identify those ports that media channels use and opens additional channels on a dynamic basis. The media and media control channels for audio and video are not inspected or monitored, because these channels only transport data and cannot open additional channels. This maximizes router and network performance to assure proper delivery of time-sensitive data.

In a specific embodiment, the firewall engine and associated application modules contain the intelligence to know when to watch payloads for new channels. When a port negotiation command is detected, the firewall recognizes that a data channel defined by port numbers for the communicating nodes is about to be established. In the payload, it detects the identity of one port of the data channel through which data will be transmitted in the a first direction. At this point, the firewall of this invention creates an ACL item for that port. The other port number of a new channel is identified in the header of the first packet following a port negotiation command that specifies the negotiated port. That packet is allowed through, and from its port number, the firewall creates another ACL item that together with the previously created ACL item defines the new data channel.

The process of overall process of parsing a payload (step **512** of process **418**) is detailed in FIG. **8**. As shown there, the process begins at **802** (corresponding to decision step **508** of process **418**) and follows with a decision step **804** in which the firewall determines whether the current session is one of the following protocols: FTP, TFTP, RPC, or SMTP. If so, it examines the payload to determine whether it has a specified intrusion signature. See decision step **806**. For these protocols, it is understood that certain intrusion mechanisms are known and used to defeat network security. These mechanisms leave certain signatures that can be specified for detection ahead of time. If such intrusion signature is identified at decision step **806**, the firewall drops the current packet and resets the connection at a step **808**.

Assuming that no intrusion signature is located at decision step **806** or that the protocol of the current packet is not one of FTP, TFTP, RPC, or SMTP, the firewall next determines whether it is expecting additional channels to be opened at a decision step **810**. As mentioned, certain types of applications may have multiple channels: typically a control channel and one or more side or data channels. The ports associated with such side or data channels cannot be known ahead of time. At step **810**, the system determines whether the packet is associated with an application that could open multiple channels (e.g., FTP or H.323) and, if so, whether any other channels might be opened for that application. As mentioned many applications, such as H.323, have an expected or maximum number of side channels.

Assuming that the application is of a type which may involve additional channels (and not all possible channels associated with that application have yet been opened), the firewall next examines the packet's payload to determine whether it includes a port negotiation command. See decision step **812**. Such commands indicate that a new channel is likely to be opened very soon. If the firewall does detect such command at decision step **812**, it next prepares to create a new passage for a new channel at a process step **814**. This involves creating a pre-gen SIS and associated ACL items as mentioned above. Note that at this stage the ACL items can specify source and destination addresses and possibly a destination port, but usually not a source port. The source port for the channel may be determined when a subsequent SYN packet for the new channel is received. After the appropriate preparations are undertaken, the process is complete at **816** (corresponding to step **514** of process **418**). Also, if either of decision steps **810** or **812** is answered in the negative, the process is completed at **816**.

Note that firewall only inspects packet payloads in control channels where port negotiation commands may appear. This conserves system resources. Further, only a subset of the packets in a control channel will have their payloads inspected. Most packets include a "command name" which indicates whether the payload is likely to contain port or address information such as a port negotiation command. If the command name is not of a type that could include a port negotiation command, the firewall discontinues its inspection of the payload. This further conserves resources.

3. State Information

As indicated above, the systems and methods of this invention preferably monitor the state of each channel. To accomplish this, they may create an SIS for each channel, even if there are other channels associated with the application. The firewall engine (and/or associated application modules) uses its knowledge of the expected behavior in each state to analyze packet headers and determine whether the current packet comports with what it expects in the current state. If a packet is not of the type expected given the current state, the firewall will drop it because it may be an illicit packet masquerading as a packet of a current session.

FIG. **9** depicts one example of a SIS **900** that may be used with this invention. The SIS includes various fields for use in monitoring a particular session. In SIS **900**, these fields include source and destination addresses **902** and **904** and ports **906** and **908** defining a socket pair, a protocol type **910** (e.g., UDP or TCP), a TCP state **912** (as defined by the TCP standard), a session state **914** as described above (e.g., closed, opening, open, closing for TCP; opening, open, and closed for UDP), sequence information **916** including the sequence numbers of the initiator's and responder's most recent packets and the size of the sequence window for the initiator and responder, timeout information **918** specifying timestamps on the most recent packet or packets in the session and relevant timeout period, various flags **920** (for e.g. inspecting at the process level, inspecting the TCP packet order, inspecting the TCP termination sequence, inspecting Network Address Translation information, inspecting the payload, etc.), a list of ACL items **922** associated with the session (dynamically created), and pointers **924** to other sessions (SISs) that form part of the same application conversation. Regarding the last of these (pointers to other sessions), note that a given application conversation such as FTP or H.323 may have multiple channels (each defined by a separate session or TCP connection). The firewall often needs to check on the status of a related session in order to make a decision about a packet in a different session.

While not illustrated in FIG. **9**, the SIS may also include alternative addresses and port numbers that may be used with a local network employing Network Address Translation. Network Address Translation (NAT) enhances network privacy by hiding internal addresses from public view. It also reduces cost of Internet access by enabling conversation of registered IP addresses. Network Address Translation is described in K. Egevang and P. Francis, "The IP Network Address Translator (NAT)," RFC 1631, Cray Communications, NTT, May 1994 which is incorporated herein by reference for all purposes.

4. Example

FIGS. **10**A through **10**C illustrate how the present invention may be employed to control an FTP session. In these

figures a router/firewall **1001** connects a local network **1003** to an external network **1005** (e.g., the Internet). Initially, router/firewall **1001** has received the SYN, ACK/SYN, and ACK packets necessary to establish an FTP control channel. All such packets had to meet criteria specified in an ACL **1007**. Upon receipt of the SYN packet, firewall/router **1001** created an SIS **1009** for the FTP control channel.

As shown in FIG. **10**A, a packet **1011** from external network **1005** enters firewall/router **1001** through an interface and must have its header checked against ACL **1007**. If it does not meet the specified by ACL **1007**, it is dropped as indicated by arrow **1013**. As the packet is for the FTP control channel, it must also meet criteria associated with SIS **1009** (state, sequence number, etc.). If it does not meet these criteria, it is dropped as indicated by arrow **1015**. In this case, it passes as indicated by arrow **1017**. Along the way, SIS **1009** is updated with information from packet **1011** (sequence number, state, etc.).

Next, as illustrated in FIF. **10**B, an FTP packet **1019** with a port negotiation command is received from local network **1003**. Because it contains a port negotiation command, firewall/router **1001** opens a pre-gen SIS **1021** to prepare for the new data channel. It also adds appropriate ACL items to ACL **1007** in anticipation of the new data channel. These items specify a first port number for the new data channel as identified in with the port negotiation command. This allows return traffic over the new channel.

Then, as illustrated in FIG. **1** OC, the first data packet for the new channel (packet **1023**) arrives from external network **1005**. Because ACL **1007** has been modified to allow it through, it passes to inspection per pre-gen SIS **1021**, which is now converted to a regular SIS. In addition, the second port number for the data channel appears in the header of packet **1023**. This information is used to modify the appropriate item(s) in ACL **1007** pertaining to the FTP data channel.

Data for the FTP data channel continues to flow across firewall/router **1001** so long as it meets the various requirements of ACL **1007** and SIS **1021**. Eventually, the connection associated with the channel is terminated and SIS **1021** is removed. The dynamically created ACL items associated with the channel are also removed from ACL **1007**.

5. Other Embodiments

Although the foregoing invention has been described in some detail for purposes of clarity of understanding, it will be apparent that certain changes and modifications may be practiced within the scope of the appended claims. For example, the local network described above may be a single local area network or multiple local area networks connected as a wide area network. Further, the security algorithm described above may be applied to a single machine as well as a network.

What is claimed is:

1. A method, implemented on a dedicated network device which receives and transmits network traffic, for limiting access to a local network, the method comprising:

receiving a packet at the network device;

identifying an application associated with the packet;

determining whether to examine the payload of the packet based on whether certain conditions are met; and

examining the packet payload based on the determination.

2. The method of claim **1**, wherein determining whether to examine the payload comprises determining whether the payload may contain an intrusion signature.

3. The method of claim **1**, wherein determining whether to examine the payload comprises determining whether the packet is an FTP packet, an RPC packet, a TFTP packet, or a SMTP packet; and

wherein examining the packet payload identifies the presence or absence of an intrusion signature.

4. The method of claim **1**, wherein determining whether to examine the payload comprises determining whether an additional channel of unknown port number may be opened.

5. The method of claim **4**, wherein examining the packet payload comprises examining the payload to identify a port negotiation command.

6. The method of claim **5**, further comprising modifying the network device to allow packets associated with the additional channel to pass.

7. The method of claim **6**, wherein the packets are allowed to pass by dynamically modifying an access control list to create a path for the additional channel.

8. The method of claim **1**, further comprising:

examining the packet's header; and

determining whether information in the packet header corresponds to an access control item.

9. The method of claim **8**, further comprising dynamically adjusting a list of access control items based upon examination of the packet payload.

10. The method of claim **1**, further comprising:

identifying a session associated with the packet;

determining whether the packet has been received after a predetermined time out period has elapsed since the last packet of the session was received; and

if the predetermined time out period has elapsed, rejecting the packet.

11. A computer program product comprising a computer readable medium on which is stored program instructions for a method, implemented on a dedicated network device which receives and transmits network traffic, the method limiting access to a local network, and comprising:

receiving a packet at a network device;

identifying an application associated with the packet;

determining whether to examine the payload of the packet based on whether certain conditions are met; and

examining the packet payload based on the determination.

12. The computer program product of claim **11**, wherein the instructions for determining whether to examine the payload comprise instructions for determining whether an additional channel of unknown port number may be opened in the application associated with the packet.

13. The computer program product of claim **11**, wherein the program instructions further specify:

identifying a session associated with the packet;

determining whether the packet has been received after a predetermined time out period has elapsed since the last packet of the session was received; and

if the predetermined time out period has elapsed, rejecting the packet.

14. A dedicated network device which receives and transmits network traffic and capable of controlling access to a local network, the network device comprising:

multiple interfaces configured to connect with distinct networks or network segments;

a memory or memories configured to store (i) one or more access control criteria for allowing or disallowing a packet based upon header information and (ii) information specifying an application conversation; and

a processor configured to compare packet header information with the access control criteria and could determine whether to examine packet payloads based upon the context of the application conversation.

15. The network device of claim **14**, wherein the network device is a router or a switch.

16. The network device of claim **14**, wherein the memory is configured to store the access control criteria in the form of an access control list.

17. The network device of claim **14**, wherein the memory is configured to store the state of at least one of a TCP session and a UDP session.

18. The network device of claim **14**, wherein the memory is configured with information specifying the context of an application conversation indicating whether a side channel may be opened for the application.

19. The network device of claim **14**, wherein the processor is configured to examine packet payloads when context information in the memory indicates that a side channel may be opened.

20. The network device of claim **19**, wherein the processor is configured to dynamically modify the access control criteria when a new side channel opens.

21. The network device of claim **14**, further comprising an operating system controlling the network device to perform functions necessary to control access to the local network and route network traffic.

22. The network device of claim **14**, wherein the network device comprises at least two processors, at least one of which is associated with one of the multiple interfaces.

23. A method implemented on a computer or dedicated network device for controlling access to a local network, the method comprising:

receiving a packet;

determining whether the packet possesses a predefined source or destination address or port;

determining whether the packet meets criteria for a current state of a TCP or UDP session with which it is associated;

determining whether to examine the packet's payload based on whether certain conditions are met; and

examining the packet's payload based on the determination.

24. The method of claim **23**, further comprising determining whether the packet sequence number falls within a defined sequence window.

25. The method of claim **23**, further comprising;

determining whether the packet has been received after a predetermined timeout period has elapsed since the last packet of the session was received; and

if the predetermined timeout period has elapsed, rejecting the packet.

26. The method of claim **23**, wherein determining whether the packet possesses the predetermined source or destination address or port comprises matching information in the packet header against information in an access control list.

27. The method of claim **23**, wherein determining whether the packet meets criteria for a current state comprises determining whether any state transition associated with a TCP or UDP session follows an expected sequence of state transitions.

28. The method of claim **23**, wherein determining whether to examine the payload comprises determining whether the payload may contain an intrusion signature.

29. The method of claim **23**, wherein determining whether to examine the payload comprises determining whether the packet is an FTP packet, an RPC, a TFTP packet, or a SMTP packet; and

wherein examining the packet payload identifies the presence or absence of an intrusion signature.

30. The method of claim **23**, wherein determining whether to examine the payload comprises determining whether an additional channel of unkown port number may be opened.

31. The method of claim **30**, wherein examining the packet payload comprises examining the payload to identify a port negotiation command.

32. The method of claim **31**, further comprising modifying the network device to allow packets associated with the additional channel to pass.

33. The method of claim **32**, wherein the packets are allowed to pass by dynamically modifying an access control list to create a path for the additional channel.

34. The method of claim **31**, wherein the packet initiates a new session, the method further comprising:

creating a state entry for the new session; and

creating one or more access control items allowing passage of packets from a node identified in the packet initiating the new session.

35. A computer program product comprising a computer readable medium on which are stored computer program instructions for a method of controlling access to a local network, the computer program instructions specifying;

receiving a packet;

determining whether the packet possesses a predefined source or destination address or port;

determining whether the packet meets criteria for a current state of a TCP or UDP session with which it is associated;

determining whether to examine the packet's payload based on whether certain conditions are met; and

examining the packet's payload based on the determination.

36. The computer program product of claim **35**, wherein the instructions for determining whether the packet meets criteria for the current state comprises instructions for determining whether any state transition associated with the TCP or UDP session follows an expected sequence of state transitions.

37. The computer program product of claim **35**, wherein the program instructions further specify;

determining whether the packet initiates a new session;

creating a state entry for the new session; and

creating one or more access control items allowing passage of packets from a node identified in the packet initiating the new session.

* * * * *

US005315657A

United States Patent [19]

Abadi et al.

[11] Patent Number: 5,315,657

[45] Date of Patent: May 24, 1994

[54] COMPOUND PRINCIPALS IN ACCESS CONTROL LISTS

[75] Inventors: **Martin Abadi**, Palo Alto, Calif.; **Andrew C. Goldstein**, Hudson; **Butler W. Lampson**, Cambridge, both of Mass.

[73] Assignee: **Digital Equipment Corporation**, Maynard, Mass.

[21] Appl. No.: **589,923**

[22] Filed: **Sep. 28, 1990**

[51] **Int. Cl.**[5] **H04L 9/32**; G06F 13/14

[52] **U.S. Cl.** **380/25**; 380/4; 380/23; 380/30; 380/49; 340/825.31; 340/825.34

[58] **Field of Search** 380/3, 4, 23-25, 380/49, 50, 21, 43; 364/222.5, 286.4, 286.5, 240.8, 246.6, 283.3, 709.5; 340/825.31, 825.34

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,309,569	1/1982	Merkle .	
4,405,829	9/1983	Rivest et al. .	
4,771,459	9/1988	Jansen .	
4,779,224	10/1988	Moseley et al. .	
4,825,354	4/1989	Agrawal et al.	364/200
4,858,117	8/1989	Di Chiara et al.	364/200
4,882,752	11/1989	Lindman et al.	340/825.34 X
4,887,077	12/1989	Irby, III et al. .	
4,919,545	4/1990	Yu	340/825.34 X
4,961,224	10/1990	Yung et al.	380/25
4,962,449	10/1990	Schlesinger	364/200
4,984,272	1/1991	McIlroy et al.	380/25
5,012,515	4/1991	McVitie	380/49

OTHER PUBLICATIONS

Schroeder, Birrell & Needham, *Experience With Grapevine: The Growth of a Distributed System*, 2 ACM Transactions on Computer Systems 3-23 (1984).

Millerm Neuman, Schiller & Saltzer, *Kerberos Authentication and Authorization System*, Project Athena Technical Plan (1987).

European Community Manufactures Association (ECMA), *Security in Open Systems-Data Elements and Service Definitions: "Alice in Wonderland"* (Jul. 1989).

EMCA, *Security in Open Systems-A Security Framework*, EMCA TR/46 (1988).

The Digital Distributed System Security Architecture, (1989).

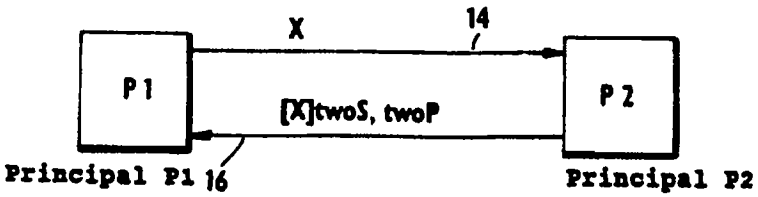
Primary Examiner—Bernarr E. Gregory

Attorney, Agent, or Firm—Arnold, White & Durkee

[57] **ABSTRACT**

An access control list for determining the access rights of principals in a distributed system to a system resource is disclosed wherein the access rights of a specified principal are based on the access rights delegated to that principal.

8 Claims, 9 Drawing Sheets



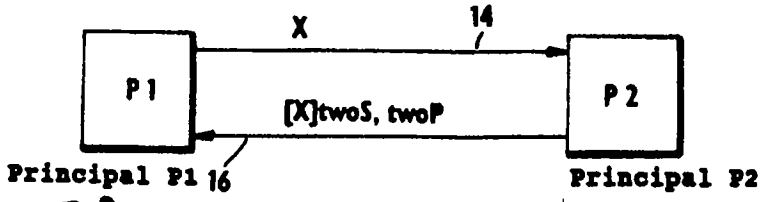


Fig. 1A

Fig. 1B

DOES X = ([X] twoS) twoP

YES → P2 HAS KNOWLEDGE OF two S

NO → P2 DOES NOT HAVE KNOWLEDGE OF two S

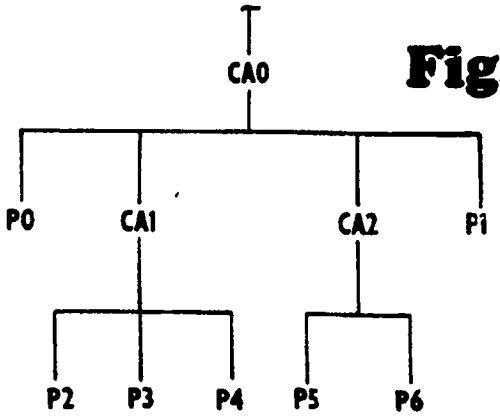


Fig. 2A

Fig. 2B

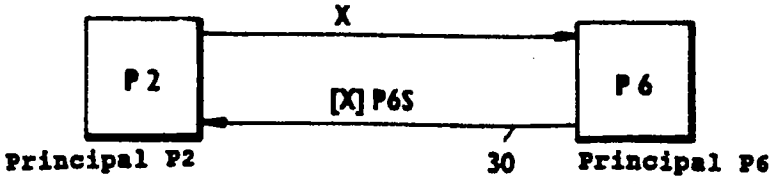
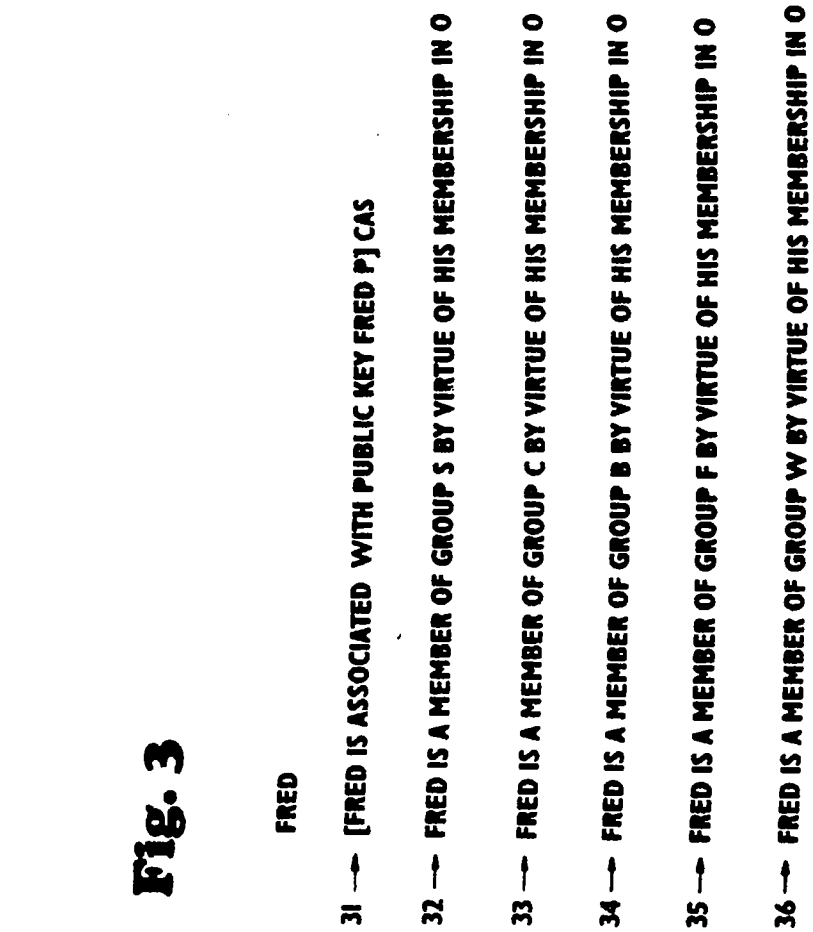


Fig. 3


FRED
31 [FRED IS ASSOCIATED WITH PUBLIC KEY FRED P] CAS
32 FRED IS A MEMBER OF GROUP S BY VIRTUE OF HIS MEMBERSHIP IN O
33 FRED IS A MEMBER OF GROUP C BY VIRTUE OF HIS MEMBERSHIP IN O
34 FRED IS A MEMBER OF GROUP B BY VIRTUE OF HIS MEMBERSHIP IN O
35 FRED IS A MEMBER OF GROUP F BY VIRTUE OF HIS MEMBERSHIP IN O
36 FRED IS A MEMBER OF GROUP W BY VIRTUE OF HIS MEMBERSHIP IN O

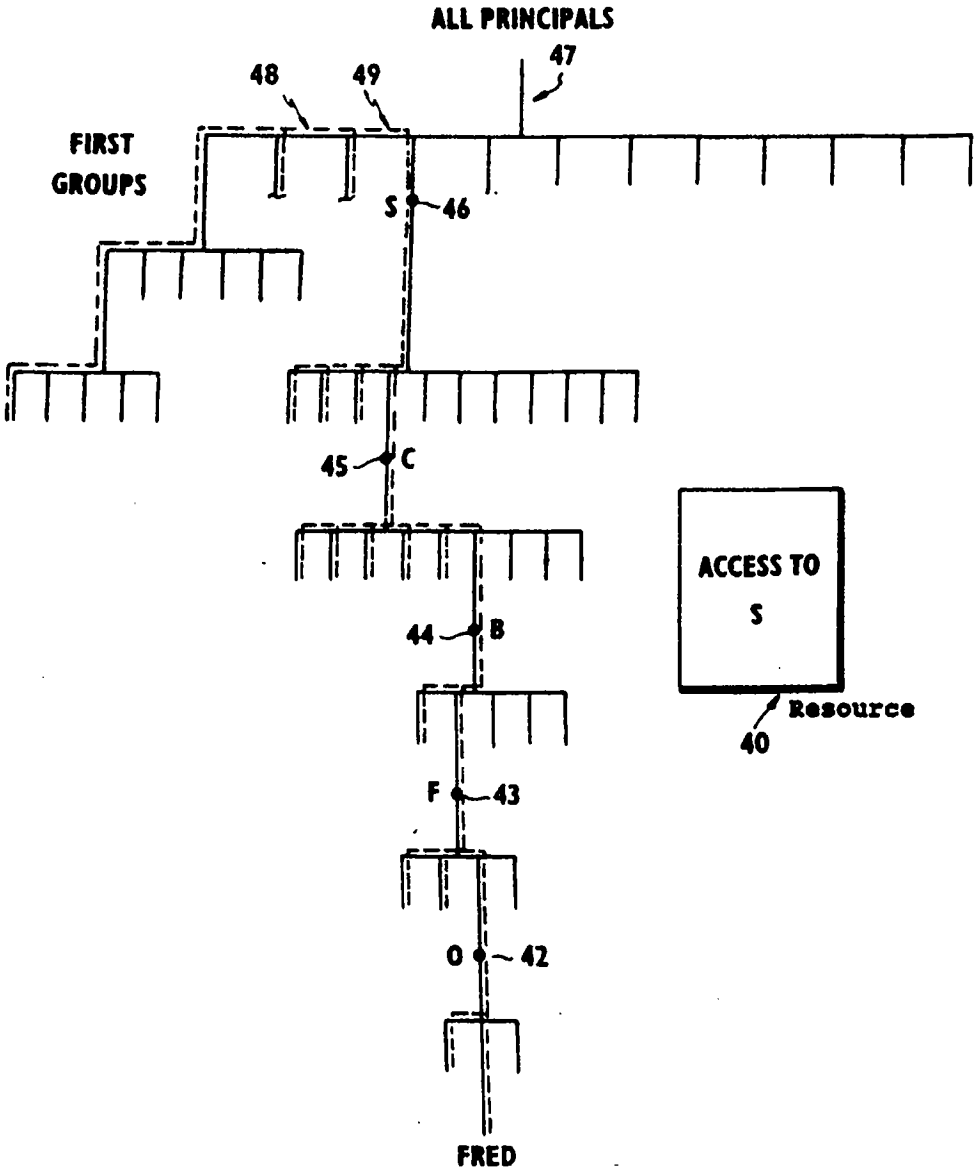

ALL PRINCIPALS
47
48
49
FIRST
GROUPS
S
46
45
C
ACCESS TO
S
44
B
Resource
40
F
43
O
42
FRED

Fig. 4A

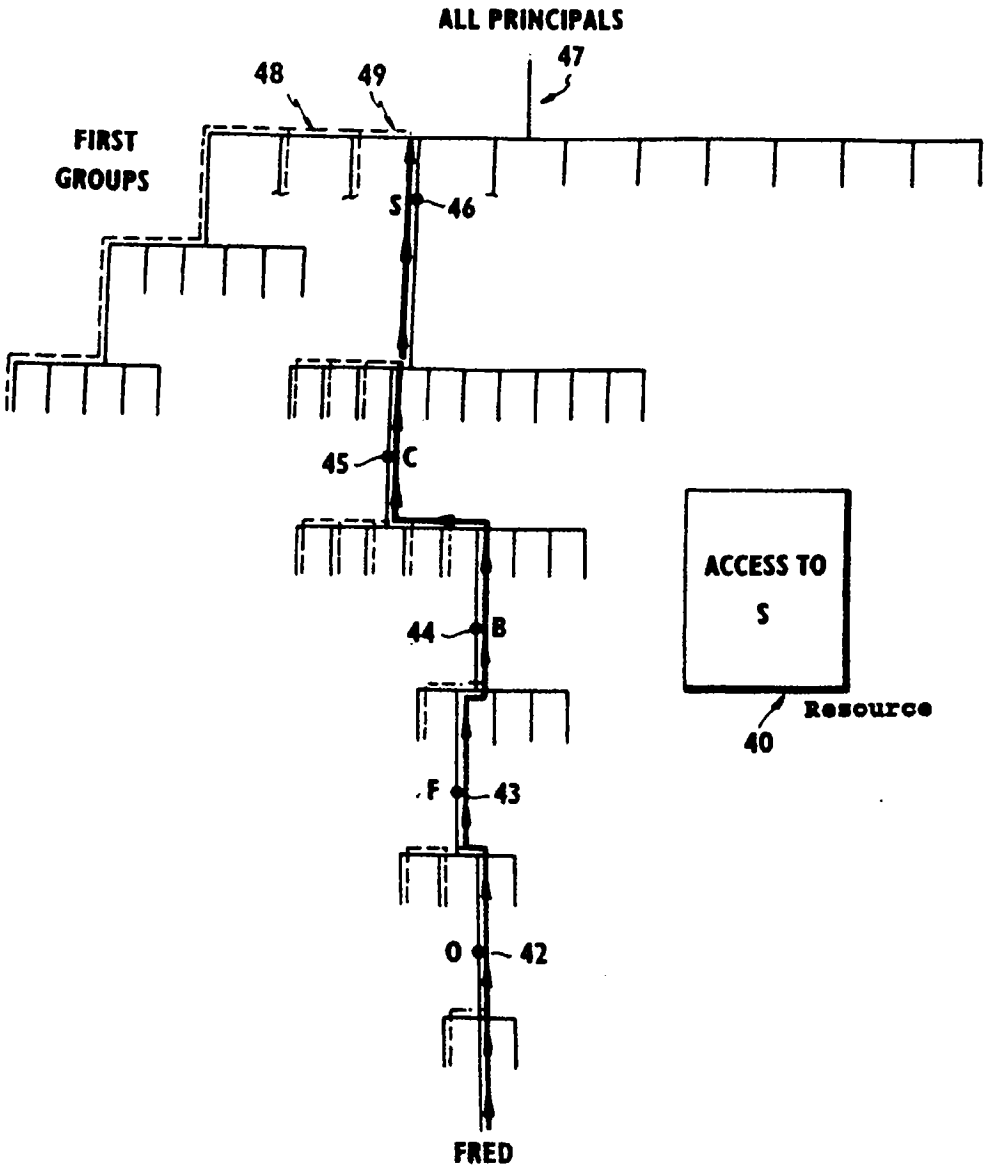
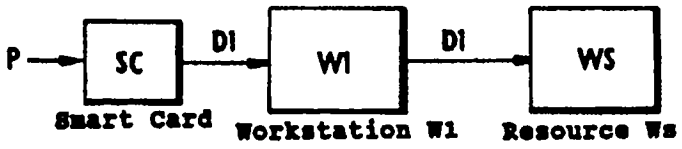
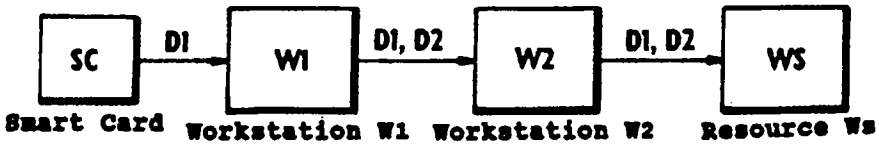

ALL PRINCIPALS
47
48
49
FIRST
GROUPS
S
46
45
C
ACCESS TO
S
Resource
40
44
B
F
43
O
42
FRED

Fig. 4B



54 → D1=[W1 FOR P] SCS

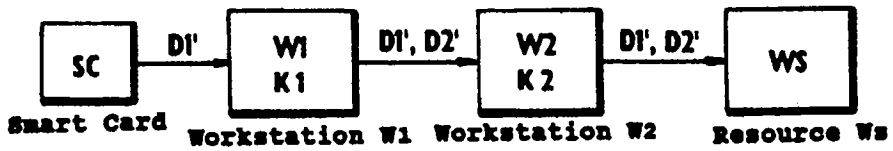
Fig. 5



54 → D1= [W1 FOR P] SCS

56 → D2=[W2 FOR P] W1S

Fig. 6



D1'=[W1 DEL P AS W1 FOR P] SCS

D2'=[W2 DEL P AS W2 FOR P] W1 DEL S

Fig. 7A

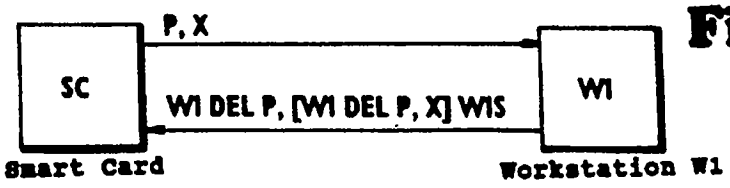


Fig. 7B

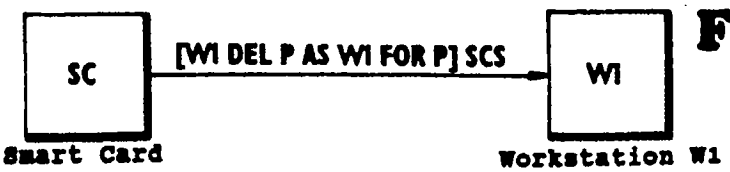


Fig. 7C

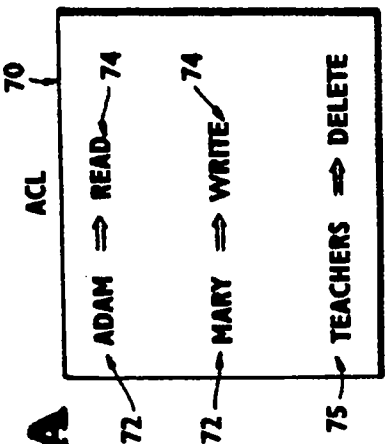


Fig. 9A

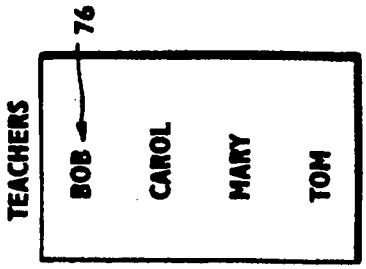


Fig. 9B

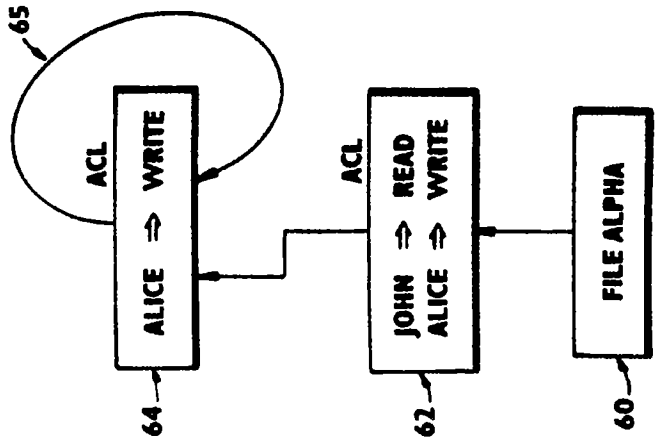


Fig. 8

```
ACL ::= ACCESS-RIGHT-EXPRESSION
        ACCESS-RIGHT-EXPRESSION AND ACL

ACCESS-RIGHT-EXPRESSION ::=
 80 → PRINCIPAL-SET  ⊨⇒  ACCESS-RIGHT ← 81
       PRINCIPAL-SET  >   ACCESS-RIGHT ← 81'
 82 → PRINCIPAL-SET

ACCESS-RIGHT ::= READ | WRITE | ...

PRINCIPAL-SET ::=
       NAME
       (ACCESS-RIGHT-EXPRESSION)
       PRINCIPAL-SET  AND  PRINCIPAL-SET
       PRINCIPAL-SET  OR   PRINCIPAL-SET
       PRINCIPAL-SET  -    NAME
       PRINCIPAL-SET  -    C(NAME)

OPERATOR PRECEDENCE:

1. -
2. OR
3. AND
4. >
5. ⊨⇒
```

Fig. 10

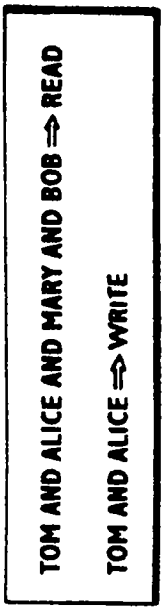


Fig. 13A

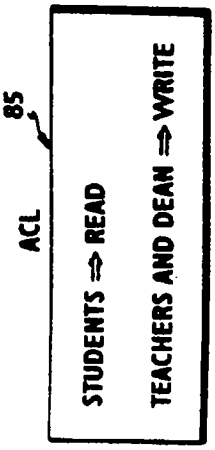


Fig. 13B

Fig. 11

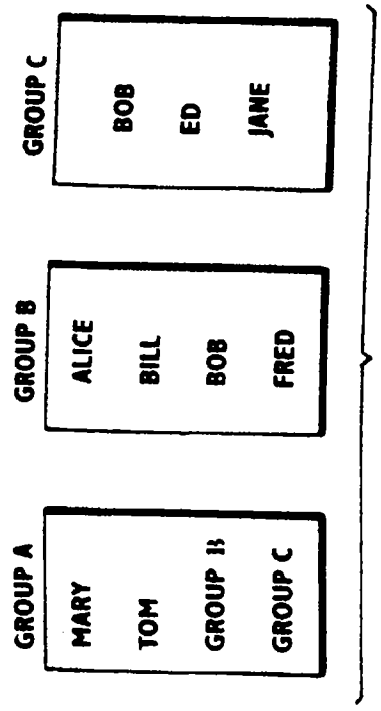


Fig. 12

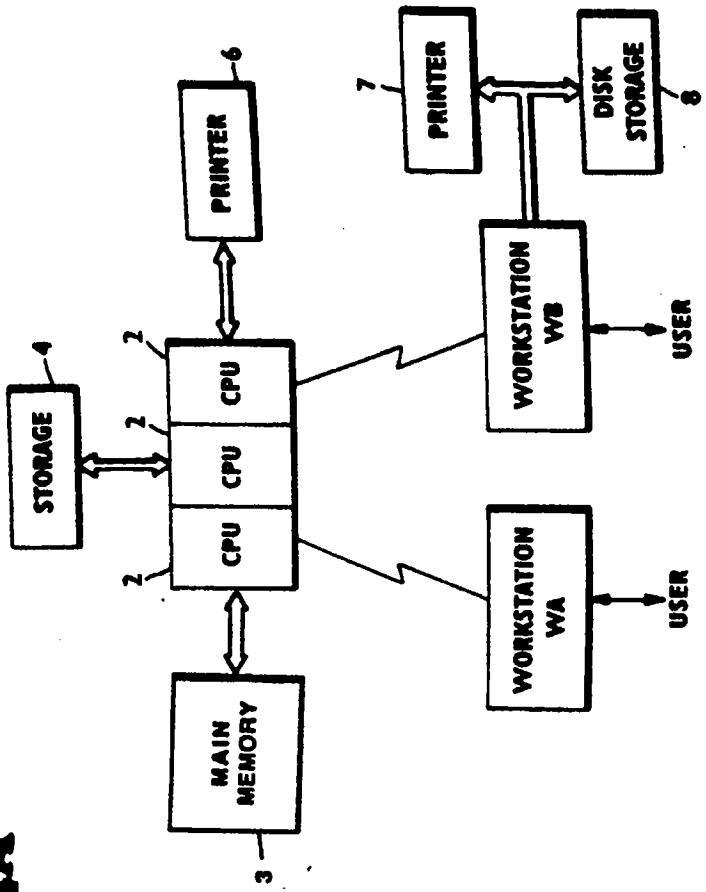


Fig. 15

Fig. 14A

ONE AND TWO ⇒ A
ONE ⇒ B
ONE AND (TWO-THREE) ⇒ C

Fig. 14B

ONE
AA AND BB ⇒ C
CC
DD ⇒ A
EE ⇒ B

Fig. 14C

AA ⇒ C
BB ⇒ C
CC ⇒ A, B, C
DD ⇒ A
EE ⇒ B

COMPOUND PRINCIPALS IN ACCESS CONTROL LISTS

CROSS REFERENCE TO RELATED APPLICATIONS

This application is one of four filed simultaneously with essentially identical specifications including: U.S. Ser. No. 589,923 by Abadi, Goldstein, & Lampson, "COMPOUND PRINCIPALS IN ACCESS CONTROL LISTS"; U.S. Ser. No. 589,924 by Gasser, Goldstein & Kaufman, "A METHOD FOR PERFORMING GROUP EXCLUSION IN LARGE HIERARCHICAL GROUP STRUCTURES"; U.S. Ser. No. 589,925 by Gasser, Goldstein, Kaufman, & Lampson, "DELEGATION TO SESSION KEY"; U.S. Ser. No. 589,926 by Gasser, Goldstein, & Kaufman, "FAST MEMBERSHIP VERIFICATION IN LARGE HIERARCHIAL GROUPS".

1. BACKGROUND OF THE INVENTION

1.1. Time Sharing Systems

In most large computing systems a timesharing computing environment is implemented. As illustrated in FIG. 1C, such a system may include "resources" such as one or more central processing units (CPUs) 2 configured to share components such as main memory 3, disk or tape storage 4, and a printer 6. The system may also include user terminals such as workstations WA and WB, which in many implementations may have their own local resources such as one or more CPUs and associated main memory (not shown) as well as perhaps a printer 7 and disk storage 8. The CPU(s) 2 execute program sequences that cause the CPUs to process commands and requests transmitted by users from the workstations WA and WB in accordance with known timesharing methods.

In such an environment, the system resources are centrally managed by a trusted authority. Because the central authority controls all access to the system resources, it is often fully trusted. In other words, the central authority is designed and maintained to ensure that the security plan for the timesharing system is properly implemented. In such timesharing environments, when a "principal" on the system (e.g., a user) requests access to a system "resource" (e.g., a printer or file server) the central trusted authority determines whether the principal possesses the necessary security attributes to access the resource. If so, the trusted authority allows the access.

In these timesharing computer systems and the like, almost all access control is handled by the central trusted authority. As such, the trustworthiness of the central authority must be maintained. Because of the importance in having a trusted central authority, many prior art devices have emphasized the importance of having a single, trusted controlling authority.

1.2. Distributed Systems

In contrast to timesharing environments, there also exist "distributed systems." In distributed systems several separate computer systems are linked together in a network to share various system resources. In such systems, there is generally no single trusted central manager that can implement the security policy for the system. As such, each system resource on the network is often required to implement its own security policy.

In such distributed systems a user typically requests access to a particular system resource. That system resource is then itself responsible for determining the access rights of the requester and allowing or rejecting the requested access.

The need for each resource to enforce its own security policy often results in complexities not encountered in timesharing environments. For example, each principal (e.g., user) on a distributed system is often assigned a user name. Access to the system resources is often on the basis of the particular access rights associated with a particular user and his name.

In theory, each system resource could include a listing of all of the principals and their access rights and user names. However, such a situation is often impractical as it would require additional memory and maintenance for each resource. Further, if numerous system resources exist, the addition (or deletion) or one principal's would require the modification of numerous lists.

One alternative utilized in the prior art is to have a central list accessible to all resources on the network. Because of the need for all system resources to have access to all of the principals and their names, a list of the principals and their names is often stored in a "global naming service" (GNS). A global naming service is a system resource which contains a list of all of the principals authorized to use the system and their names. Unlike a timesharing environment where the naming service is centrally controlled, in a distributed environment, the naming service is merely one of many system resources.

In most security systems, access to a system resource is determined on the basis of group memberships. Thus, the security policy of a particular resource may dictate that members of only a certain number of selected groups have access to that resource. Because the principal requests access to the resource, the resource must determine whether the requester is a member of one of the selected groups. If so, access is allowed. If not, access is denied.

Because it may be desirable for groups to contain other groups (or subgroups) the verification of a particular principal as a member of a large structured group can often be extremely slow.

1.3. Security Needs for a Distributed System

As discussed above, security systems for distributed networks often encounter complexities not found in centralized networks. For example, any system attempting to provide security for a distributed network must have the ability for a user to authorize a computer to operate on user's behalf and only to do so while authorized. Further such a system should have the ability for an authorized computer to present authorization to other computers in a secure and verifiable manner; and the ability for the user to rescind the authorization.

Because distributed systems generally have several workstations, it is desirable to allow a user to access the system resources regardless of which workstation he is logged into. However, because all systems on the network cannot be equally trusted, it may be desirable to prevent a user from accessing certain information on certain untrusted workstations.

Second, because distributed networks often have a large number of network entities, it is generally desirable to organize the entities into manageable groups. To implement a security policy properly, it is desirable to

be able to manage these groups through an effective security policy.

2. SUMMARY OF THE INVENTION

The present invention addresses the goals discussed above through the use of a unique distributed security system. Each entity on the distributed is given a unique name and a private key. This private key enables each entity to identify itself to other entities and to encode certain messages. The messages encoded by the private keys may be decoded through the use of public keys. The public keys are stored with the entities names in a global naming service.

The global naming service includes a list of each of the entities, and a listing of the groupings and subgroupings into which the entities are divided. A group hint may be stored with the entities' name to enable fast searching of large groups.

Each entity may delegate access rights to other entities on the network. To prevent improper use of delegation, the present invention provides for the generation of different session keys each time the user logs onto or off of the network.

A unique method for implementing security policies is also disclosed. An access control list is provided for each system resource. This access control list contains a list of all possible access privileges and the user's that have these privileges. Thus, when a user requests access to a system resource, the user's name is compared to the resource's access control list. If the user's name if found on the list next to the requested access, access will be granted.

The methods, techniques, and data structures described here may be implemented by loading one or more sequences of program statements from a program storage device (e.g., a computer tape, a magnetic or optical disk storage device, or other storage means) into the memory or memories of a digital computer or computers configured as described herein and causing the computer(s) to execute program sequences as appropriate. The design and development of specific program sequences is a matter of routine work for persons of ordinary skill (which of course should include familiarity of the computer(s), operating system(s), etc. that are selected for the implementation) who have the benefit of this disclosure.

3. BRIEF DESCRIPTION OF THE DRAWINGS

FIGS. 1A-1B illustrate the use of encryption keys to authenticate principals in a distributed computing system.

FIG. 1C illustrates a timesharing computing environment.

FIGS. 2A-2B illustrate a method for authenticating principals in the present invention.

FIGS. 3, 4A and 4B illustrate the use of search hints in the present invention.

FIG. 5 illustrates one method for delegating authorization in the present invention.

FIG. 6 illustrates chained delegation.

FIGS. 7A-7C illustrate delegation to a session key.

FIGS. 8 and 9A-9B illustrate the use of access control lists.

FIGS. 10 and 11 illustrate basic access right expressions.

FIG. 12 illustrated group exclusion in the present invention.

FIGS. 13A-14C illustrate the use of compound principals in access control lists.

FIG. 15 illustrates a timesharing computing environment.

4. DEFINITIONS AND CONVENTIONS

This specification discloses aspects of a distributed security system in which access to system resources is controlled by access control lists associated with each system resource. When a user makes a request of a resource, the reference monitor (i.e., the manager of access to the resource) for that resource looks for the requesting user on that resource's access control list. If the user's name is found (or membership in a group is verified), the requested access is granted.

Because the system of the invention deals with encoded and decoded messages and with public and private (secret) keys, a convention is adopted for ease of illustration. The public key of a principal is represented as xxxxP, where xxxx is the name of the principal, and the capital P indicates that it is a public key. Conversely, the private or secret key for a principal is represented as xxxxS, where xxx is the name of the principal, and S indicates that it a private or secret key.

Brackets [] are used to indicate that a message has been encoded. The key that has been used to encode a particular message may be represented to the right of the closing bracket. For example, if a message YYY is encoded using the private key of principal david, the encoded message is represented as [YYY] davidS.

The message obtained when an encoded message is decoded is represented through the use of parenthesis, ( ). The encoded message to be decoded is represented within the parenthesis, while the key used to perform the decoding is represented to the right of the closing parenthesis. For example, if the example message above is to be decoded using the public key of david, the resulting message is represented as ([YYY] davidS) davidP. Because the key davidP is the complement to davidS, the decoded example will be equivalent to the original, uncoded message. In other words, YYY=([YYY] davidS) davidP.

In the specification and claims that follow, the term "workstation" is used in its generic sense and, as such, describes various types of computer systems.

5. AUTHENTICATION

In order to implement a security policy controlling the exchange of information throughout a distributed system some mechanism should exist for uniquely identifying each of the network systems. Only in this manner can the access rights of each system be determined and controlled. This process of identifying and verifying a principal on the network, is known as "authentication." In an embodiment of the present invention, authentication is accomplished through the use of RSA cryptography and a global naming service.

The authentication process illustrated here is a two-part process. First, a user seeking to authenticate itself must demonstrate knowledge of a particular private key. Second, the entity receiving the authentication request must accurately be able to determine that knowledge of a particular private key implies a particular principal name. These two components of authentication are kept separate in the present invention.

5.1. RSA Cryptography for Verifying Knowledge of a Particular Private Key

RSA cryptography, disclosed in U.S. Pat. No. 4,405,829 to Rivest et al., is well known in the art. RSA cryptography involves the use of a public/private key system.

In connection with the present invention, each principal on the network is assigned a particular "private key". This private key is a code that is exclusive to that principal; it is not disclosed to any other principal on the network. Thus, for security purposes, it is assumed that each private key is kept secret by its principal. For this reason private keys will be referred to in this specification and drawings as xxxS, where xxx identifies the key, and S indicates that it is a private (or Secret) key.

Corresponding to each private key is a public key (to be represented as xxxP). A public key is associated with each principal on the network. This public key may be made known, and shared with other principals on the network.

As known in the art, and described in the Rivest et al. patent, the public and private keys are generated in such a manner that knowledge of the public key does not reveal the private key.

The public and private keys operate together to allow the coding and decoding of messages. Thus, a message encoded using a private key may only be decoded by the public key that corresponds to that private key. Alternately, a message encoded using a public key may only be decoded by the private key corresponding to that public key.

FIG. 1A provides an example of using RSA encoding for authentication. P2 establishes a communications channel over which it wishes to converse with P1. If principal P2 wishes to converse with principal P1 it must first persuade principal P1 that it is indeed principal P2.

This may be done by principal P1 generating a challenge to principal P2. This challenge (sometimes referred to in the art as a "nonce") comprises a random number X that is sent to principal P2. This nonce message is illustrated as **14**.

After receiving this nonce **14**, principal P2 sends to P1 the value obtained when the number X is "signed" (encoded) by principal P2's private key (twoS). It also includes in this message its corresponding public key twoP. This message is illustrated as shown at reference numeral **16**. As may be discerned, the random number X is illustrated as being "locked" by the key twoS. This illustrative convention illustrates that the message within the brackets has been signed (encoded) by the key at the right of the bracket.

Thus, the message may only be decoded (unlocked) by using the key complementary to that outside of the brackets. For example, in FIG. 1A, message **16** shows the number X encoded by the key twoS. As such, this message may only be decoded through the use of twoP (the complement to twoS).

After receiving message **16**, principal P1 can compare the uncoded number X (from message **14**) with the value obtained when the signal **16** is decoded using the public key twoP (which was provided in the signal **16**). If the two values match, then principal P1 knows with certainty that it is communicating with a principal possessing the private key twoS. Because P1 has chosen the value of the challenge X, it is not possible for an imposter to impersonate P2 by replaying a response from the real P2 captured from a previous exchange. This process is illustrated in FIG. 1B. This is the first part of the authentication process.

If principal P1 can somehow associate the private key twoS with principal P2, then principal P1 knows that it is communicating with principal P2, inasmuch as no one else could have signed number X with principal P2's private key.

A similar exchange in the opposite direction allows P2 to authenticate P1. Once each principal is assured of who he is talking to, messages may be sent over the communications channel. Note that to prevent an imposter from subsequently breaking into the conversation, the communications channel must be secured by means generally known, such as physical security or conventional secret key encryption.

In the above example, principal P2 provided its public keytwoP in message **16**. In one embodiment of the invention principal P2 provides only its user name. In this embodiment, principal P1 would have been responsible for determining the public key associated with the user name sent in the message.

5.2. Improved Global Naming Service

The above discussion of authentication placed particular emphasis on the relationship between a particular principal and the public and private keys associated with that principal. Each principal on the distributed network of the present invention is assigned a name and a private key.

As discussed above, in order to determine which principal is requesting authentication, the non-requesting party must (1) determine that the requesting party has knowledge of a private key, and (2) determine which particular principal is identified by that private key.

In connection with the present invention, principals are identified through the use of "principal names." A principal name is an identification code that uniquely identifies a principal. In one embodiment, the principle names are human readable and understandable so that people can specify this nameon access control lists. Because the security system of the present invention is geared toward principal names, not private keys, some mechanism should exist for associating the public and private keys for each principal with its name.

This function is accomplished through the use of an improved global naming service (GNS). This naming service is a depository of principal names, their associated public keys, and other relevant security information. This information (the principal names, public keys, etc.) is stored in name "certificates" that are signed (encoded) using the private keys of particular principals that have some degree of trust. Because these principals certify that a particular principal has particular access rights, they are sometimes referred to as certifying authorities (CAs).

(a) Trust and the Improved Global Naming Service. The improved GNS need not be fully trusted. This is beneficial in that is simplifies the design and maintenance of the GNS. Further, the overall security of the system is increased because many different servers do not have to be trusted.

For purposes of the present invention, the GNS should be trusted only for "rapid revocation." This means that if a key is compromised, the global naming service can be trusted to delete the naming certificate if requested to do so by the appropriate principal. Meth-

ods for developing such a limited trust GNS are known in the art and will not be discussed herein.

(b) Principal Names: Two Basic Types. The named principals contained in the global naming system may be of at least two types: users and systems. A "user" is defined as the person who uses systems and resources, and instigates access to "objects." A "system" is basically defined as a state machine (i.e., a device that when given a current state, a translation function, and some inputs yields a new state and a set of outputs). In simple terms, a system is a computer running a piece of specific software. A system may be a computer running a certain piece of software, or a process running underneath another system (e.g., a process running within a particular operating system). Systems having other systems running underneath them are sometimes referred to as "engines."

5.3. Certifying Authorities

As briefly alluded to above, the global naming service of the present invention actually contains certificates that are encoded (signed) using the private keys of special principals known as certifying authorities (CAs). Thus, any principal possessing knowledge of a particular certifying authority's public key may decode and read these certificates.

For example, the certifying authority for a particular system may be a group manager who is normally not connected to the network. When the network manager wishes to add a principal to the GNS, he may go "online" onto the network, use his private key to encode the information concerning the new principal (e.g., the principal's name and public key), and deposit this encoded message into the GNS. In this manner, anyone having the public key of the network manager may access the added certificate (which contains the new principal's public key), determine that a new principal has been added to the group, and attempt to access (or receive access requests from) the new principal.

(a) Multiple Certification Authorities; Certification Authority Hierarchy. For large networks, more than one certifying authority may exist. Multiple certifying authorities may be used to ease management concerns and improve performance and security. In the present invention, these multiple certifying authorities are organized into a specific certification authority hierarchy.

When multiple certifying authorities are used, the global naming service may be divided into several directories and subdirectories. Each directory should contain certificates (signed by the certifying authority for that directory) for the principals and subdirectories within the directory; further, each directory should identify the parent directory's certifying authority. In addition to the multiple directories, when multiple certifying authorities are used, each principal should maintain both its private key (which it keeps secret), and the public key of the certifying authority for the directory in which it is named. Generally, the certifying authorities at the nearby branches of the directory will be trusted more than those controlling more remote directories. Since the structure of the namespace is expected to reflect the structure of the organization using it, this naturally reflects the greater trust one places in more closely related organizations.

Thus, when two users not certified by the same certifying authority wish to mutually authenticate, the various certification authorities must cross-certify each other. This requires the certifying authorities to identify each other up to a common ancestor.

FIG. 2A illustrates one example of such authentication. As illustrated there are seven principals P0–P6. Each of the principals has a naming certificate stored in the global naming service which is signed by the certifying authority for the directory in which it is found. For example the global naming service entry for principal P5 would contain a naming certificate signed by certifying authority CA2.

If principal P6 wished to authenticate itself to principal P2, P2 could send a message containing a nonce (e.g., an uncoded randomly generated number) to P6. P6 then returns the nonce encoded under its private key P6S. Such a message is illustrated at reference numeral 30 in FIG. 2B. After receiving this message, principal P2 can look up the certificate for principal P6. However, this certificate is signed by CA2; P2's certification authority is CA1, so P2 has no a priori means of verifying the certificate. P2 solves this problem as follows:

P2 knows the public key of his CA, CA1P. In the naming service, he finds the certificate signed by CA1 that identifies CA1's parent to be CA0 and its public key to be CA0P. He verifies this certificate by decoding it with CA1P.

P2 then looks up the certificate signed by CA0 that certifies CA2. He decodes this with CA0P obtained above and learns that CA2's public key is CA2P. Finally, P2 can decode the certificate that associates P6 with its public key P6P and verify that P6 correctly encoded the nonce with P6S.

Because the certifying authority for CA1 trusts certifying authority CA0, and likewise certifying authority CA0 certifies certifying authority CA2, it follows that principal P2 can then trust certifying authority CA2.

(b) Contents of an Authentication Certificate. In its simplest form, an authentication certificate contains the name of a principal, its public key, and a time period of validity. (When multiple certifying authorities are used, the certificates may also contain the name of the certifying authority.) As discussed above, these authentication certificates are signed by a certification authority and are stored in the global naming service.

A time period of validity is included with the authentication certificates to ensure that a compromised principal does not stay on the network indefinitely. When a authentication certificate is stored in the global naming service, so too is a period of validity. When the period of validity has expired, the authentication certificate is no longer valid. As such, the authentication certificates must be periodically updated.

6. VERIFICATION OF GROUP MEMBERSHIP

As discussed above, the global naming service contains a listing of all of the principals authorized to operate on the network. When a distributed network becomes large, this list can often become quite lengthy. A system in accordance with the present invention allows numerous users to be lumped together and treated as a single unit called a "group."

In distributed systems, multiple objects often must be accessible to the same sets of users. These sets of users can be quite small (e.g., two users) or they can be quite large (e.g., 100,000 users). In order to effect fast and manageable access, principals who are considered equivalent for security related purposes may be combined into a group. A group may be thought of as a list

of principals. The entries within a group make up the "group definition."

As discussed below, the entries in a group may be signed certificates (see section 6.2, below) or simply a list of names.

6.1. Nested Groups and Subgroup Searching

Just as a group may contain a principal name as one of its entries, so too may a group contain another group (a "subgroup"). This nesting of groups allows large groups to be expressed as trees of other subgroups.

In the absence of specific optimization (discussed below), extensive searching of a group and its subgroups must be carried out for membership verification. For example, if a group has ten subgroups, each with ten members, the possibility exists that each of the 10 subgroups must be searched before a certain principal is identified as a member of the group. Such searches may be extremely slow. While the structure of an individual group can be optimized to allow a fast search, the various subgroups are likely to be distributed over different portions of the global naming service.

6.2. Certification of Groups

Because the storage medium of a group definition may not be adequately secure, the group definitions may be certified. This is done by enclosing the group definition in a certificate, or multiple certificates signed by some appropriate certifying authority. The certifying authority should be the entity that has control over the group. The group's certifying authority (GCA) may in turn be certified by the certifying authority of the group's parent directory in the form of:

certifying authority certifies that GCA with key GCAS may sign membership certificates for group G

This format is similar to that used to certify a directory certifying authority.

For reasons of efficiency, it may be desirable to represent group membership with multiple certificates. For example, consider a group having as its members the principals P1, P2, P3 . . . pn (where P1, etc. are either individual principals or other subgroups (principal-sets)). Such a group is represented by a certificated for each principal-set, e.g.;

GCA certifies that P1 is a member of G
GCA certifies that P2 is a member of G
GCA certifies that P3 is a member of G
GCA certifies that Pn is a member of G

These certificates have the same form as the certificates that certify a principal, except that certificates that authenticate a principal associate a particular public key with a principal's name, whereas group certification keys associate a particular principal with a group.

Because membership in a group is asserted in a certificate, it is not possible for a misbehaving (or compromised) entity to forge bogus group membership. A storage device can refuse to furnish a certificate when requested, or deny its existence, but it cannot forge a bogus certificate. In most instances, failure to furnish a certificate would only deny membership in a group, and therefor deny access to some object.

While a entity could not create a bogus certificate, the possibility exists that the entity may retain a certificate that has been revoked. To reduce this risk, the group membership certificates may have a timeout period (i.e., the membership certificate expires after a predetermined time period unless re-asserted).

Some groups may be uncertified (i.e., a group stored without the benefit of certifying certificates). The groups are only as secure as the entity that controls modification access to the group list.

6.3. Problems Arising from Large Groups and the GNS

As discussed above, the names of all entities on the network are stored in a global naming service along with certain attributes for those entities. Also stored in the global naming service are the group definitions. The definition of a group is found under the naming service entry for that group's name. As noted above, this group definition consists of a listing of each of the members of the groups. The group definition may also include a listing of subgroups.

As discussed above, for security purposes, the listing of a group's members is done through the use of certificates. These certificates are messages "signed" by trusted authorities that verify group membership. The certificates which comprise a group definition (when decoded) merely provide the name of a principal (or subgroup) that is a member of that group; the security attributes are stored with the name of that principal in the global naming service.

When a principal attempts to access a system resource, the groups having access to that resource must be searched to determine if the requesting principal is a member; each subgroup in a group, and each sub-sub group must also be searched. To perform such a nested search, each of the global definitions for the sub-groups and sub-sub-groups must be accessed at different locations in the GNS. In large networks, having numerous users with numerous subgroups, this type of searching can take considerable time.

6.4. The Use of Search Hints

To improve searching speed, group membership hints may be stored with the principal's name in the global naming service in the form of:

P is a member of group G by virtue of membership in G' where G' is a subgroup of G that lists P as an explicit member. Where multiple nested subgroups exist, e.g., P is a member of G1; G1 is a member of G2; G2 is a member of . . . Gn, several hints may be stored with the principal's name in the form of:

P is a member of group Gj by virtue of his membership in G1

For j=1 . . . (n−1)

Note that the hint list is inverted order. In other words, the hints indicate P is a member of the largest group possible. This is because it is the searches of large groups that slow down the search process.

For example, suppose that a principal named FRED has an office O in a particular wing W of a particular floor F of a particular building B in a particular city C of a particular state S. Several hints could be stored with FRED's name in the global naming service as illustrated in FIG. 3.

The first entry in the global naming service for principal FRED is an authentication certificate 31 that is signed using the private key of the certifying authority CA. Also found in FRED's global naming service entry are six hints (32-37) indicating FRED's membership is certain large groups. The use of these hints greatly decreases the amount of search time needed to verify membership in a particular group. Using the same principal FRED, as in FIG. 3, FIG. 4*a* illustrates a search done without using hints. In this example, a resource 40

allows write access to all members of group S. Without the use of hints, each office of each wing of each building of each city of each state must be searched until a match for FRED is found. Specifically, each group and subgroup of branches **48** and **49** will have to be completely searched, as will many if not most of the branches of group **46**.

The search speed is greatly increased when hints are used. As illustrated in FIG. **4B**, with hints, principal **40** now has a hint that FRED is in S via his membership in group O. The reference monitor then merely has to expand group O, and verify that group O contains a membership certificate **41** for FRED. Resource **40** may also have to verify that group O is a subgroup of group S, but this can be done through the use of hints for groups as discussed below.

In the above example, hints were provided for each group of which FRED is a member. Generally, it is not beneficial to have hints for each and every subgroup. Thus, it may be desired to include hints only for the largest of groups. For example, assuming that the subgroups of group **44** are not too large, only the hints as to membership in groups S, C, and B may be provided.

For the above example, once it is determined via a hint that FRED is a member of group S via membership in group B, the search for FRED as a member of group B would be accomplished as if no hints were used.

6.5. Hints for Groups

Although described above with respect to principals, search hints may be stored for groups that are members of larger groups. For example, if a group G1 is a subgroup of G2 which is a subgroup of G3 . . . which is a subgroup of Gn, the hints may be in the form of:

G1 is a member of group Gj by virtue of membership in G2.

For $j=2 \ldots (n-1)$.

The searching procedure for such group hints is substantially the same as that used for principal hints.

6.6. Preventing Search Delays

The presence of hints reduces the search time of a large group. To avoid excessively long searches, it may be desirable for a large group to have as an attribute that a hint is required before a search will be performed. Thus, if a principal lacks the hint for an "inversion hint required" group, the membership test fails quickly.

Alternately, a maximum search effort may be defined for a group lacking hints. Once the maximum search effort is exceeded, the access attempt fails with a "group too big error."

6.7. Inversion Hints and Security

The inversion hints for a principal merely limit a search to a particular group (or subgroup). It is from this limited group that the certificate attesting to the principal's membership in the group is obtained. Because the group membership is still certified by the group certifying authority, the inversion hints do not attest to the principal's membership in any group, and do not have to be certified.

In other words, the certificates attesting to a principal's membership in a group are stored with the actual group definitions, not with the principal's name. Thus, the information provided in the him is truly only a hint and can at worst deny access if its wrong or if the principal feigns membership in a group that it isn't a member of.

Separating the principal's membership certificates from its name preserves the control needed over revocation. For example, if the membership certificates were stored with the principal's naming service then whoever had write-access to that entry could subvert group revocation by merely replacing the certificates.

6.8 Maintaining the Search Hints

Because group membership hints constitute a duplicate representation of the group memberships in the distributed environment, there some mechanism should exist by which the hints are kept in synchronization with the actual group definitions. The problem is complicated by the fact that in general the individuals managing a group may not have writing access to the naming service entries of the members. There are several possible mechanisms for keeping the hints current.

(a) Electronic Mail and Hint Maintenance. The management actions of adding or removing members from a group may result in electronic mail being sent to the affected principal, or the system manager controlling that principal's naming service entry. This mail may be interpreted to add, update, or remove the affected hints. The use of electronic mail is known in the art and will not be discussed herein.

(b) Daemon Processes to Maintain Hints. A daemon process, running under an authority which has write access to the principal's naming service entry may be used. This process may either:

(1) scan all of the large groups in the naming service and add or update the principal's hints in their naming service; or

(2) scan the hints stored with each principal and remove those whose membership certificates are not found in their respective groups.

7. HUMAN USER AUTHENTICATION

The case of authenticating a human user is special in that the human user does not have direct control over a RSA private key. That control must rest in some piece of hardware or software that the user can trust.

One possible means for authenticating a human user is a "smart card." A smart card is a piece of electronic equipment that is electrically coupled to the user's terminal and has a keypad, display, clock, and logic for performing RSA operations.

A central certifying authority (perhaps a human resource manager) may issue a smart card to a human user. This smart card, when activated, should be willing to give certain information, including the name of the user, to anyone. The smart card should also be able to generate a nonce challenge that it gives to a workstation.

To initiate a computing session, the user and workstation mutually authenticate each other. This is accomplished by having the smart card issue the user's name and a nonce challenge to the workstation. Given the user's name, the workstation can retrieve from the global naming service all of the certificates needed to authenticate the user and determine that the user can logon to that workstation. As will be discussed below, this may be accomplished by the workstation comparing the user's name to its access control list.

The workstation signs the challenge with its private key and returns to the smart card the signed challenge, another nonce, a public key, and the certificates retrieved from the naming service that the smart card needs to authenticate the workstation.

The smart card authenticates the workstation by verifying the signed challenge and the certificates. It then displays the identity of the workstation on its display. The user, satisfied with the identity of the workstation, enters his PIN on the smart card to authorize the session.

The smart card now signs a message containing the nonce and public key received from the workstation and sends this message back to the workstation. The workstation can now authenticate the user by verifying the signature on the nonce using the certificates fetched above. The public key signed by the smart card gives the workstation the authority to act on behalf of the user, as discussed below.

8. DELEGATION

8.1. Basic Principles of Delegation

Once the user has authenticated himself to the workstation, he may access any local files within the workstation to which he has access. If the user, however, needs to access remote files on a remote system resource, the user must delegate to the workstation the ability to access those files on his behalf.

Such a delegation is illustrated in FIG. 5. In the figure, a user P has been authenticated to workstation W1, but wishes to access remote files in resource Ws. Ws receives access requests, not directly from the user P, but from workstation W1. Because access rights are defined in terms of the user, the reference monitor in the server Ws must have some way of verifying that the access request did indeed originate with the user P.

This verification is accomplished through the use of "delegation certificates." The delegation from the user to the workstation is represented by a certificate, signed by the user's smart card SC at the time of login. This is represented by certificate D1 in FIG. 5. This certificate indicates that the workstation is authorized to speak for the user. The delegation format illustrated indicates that the smart card SC 53 has signed a statement authorizing W1 to speak for user P. The workstation W1 forwards this certificate to the remote resource Ws as proof of delegation.

The remote resources may then compare the user P's name, as well as the name of the delegated workstation W1, with the names of the users and delegated systems in its access control list (ACL). If the requesting user on the delegated workstation is found in the ACL, then the user, through the workstation, is granted access to the resource.

8.2. Chained Delegation

In many situations, more than one system is present between the user and the system resource. Such a situation is illustrated in FIG. 6. In that figure a user (not shown) on workstation W1 seeks accesses to a file on system resource Ws, through workstation W2. In this case, the user's smart card SC delegates to workstation W1 via delegation certificate D1, which in turn delegates to W2 via delegation certificate D2.

As discussed above, the first delegation happens where workstation W1 is authorized to speak for the user (certificate D1 in the figure). The second delegation says that workstation W1 permits W2 to speak for the user. Before making an access request to system resource Ws, the workstation W2 forwards a copy of both its own delegation certificate D1 and the workstation's delegation certificate D2.

Through a chain of reasoning using both delegation certificates and the authentication certificate, the system resource Ws can conclude that workstation W2 is indeed authorized to speak for the user P. As before, both the user's name, and the names of workstations W1 and W2 must appear on the resource Ws's access control list.

Although illustrated with only two delegations, a long delegation chain may exist between several workstations. To effect such a chain, the system resource Ws must receive and verify all of the delegation certificates.

For security purposes, the delegation certificates are only valid for a limited time period (approximately a day).

8.3. Delegation to a Session Key

(a) The Need to Terminate Delegations. One security problem with the above delegation system is that the identity of the workstations W1 and W2 are relatively permanent. Thus, once the user P authorizes workstation W1 to speak for him, the workstation W1 can make requests in user P's name as long as the delegation certificate is valid. Since the valid time period for a delegation is generally longer than most user computing sessions, there is a possibility that workstation W1, if compromised, could continue to make requests for user P even after P is logged out of the system.

This problem cannot be solved by merely erasing the delegation certificate from workstation W1's storage. The delegation certificate has been transmitted to other parties (e.g., workstation W2, system resource Ws). Thus, even if the delegation certificate was erased from workstation W1's storage, an intruder could obtain a copy from somewhere in the system (e.g., from system resource Ws), compromise the workstation W1, and begin making requests in user P's name. Under this system, a high degree of trust must be placed in workstation W1. In order to prevent unauthorized requests, workstation W1 must be protected from compromise not only while user P has an active session with workstation W1, but for all time subsequent to the initial delegation. (i.e., the time between user logout and the expiration of the delegation) To avoid the need to place such high trust in workstation W1, the present invention makes use of session or delegation key pairs.

(b) The Session Key. To allow a delegation to be rescinded at a time of the user's choosing, a system in accordance with the present invention utilizes the concept of delegating to a session. A session is simply the context in which all a user's actions are performed on a computer. In one embodiment of the present invention a new public/private key pair is generated each time the user delegates to a workstation.

As discussed above, when the user P's smart card SC is inserted into the workstation the smart card provides the user P's name, and generates a nonce challenge. In the present embodiment, after verifying that the user P can access the workstation, the workstation generates a new public/private key pair, and sends to the user's smart card the new public key, as well as a signed message containing the smart card's nonce challenge and the new public key. Note that because this public key is unique for each login, it also constitutes the nonce sent back to the smart card, which was called out separately in section 7, above. Thus, when the user's smart card receives the return message, it decodes the message using the workstation's public key and verifies that the response to the nonce challenge was correct. At this

point the user's smart card knows the new public key with which the workstation is communicating.

FIGS. 7A through 7C illustrate the use of such session keys. In FIG. 7A, the symbol K1 represents the new delegation key whose private key W1delS is held by workstation W1. After workstation W1 determines that the user P can access the workstation, it sends a message signed with workstation W1's normal private key W1S. This signed message contains the number generated by the user's smart card SC in the nonce challenge, as well as the new public delegation key W1delP.

As depicted in FIGS. 7B and 7C, the smart card SC, already knowing workstation W1's normal public key (from the certificates provided by the workstation), can decode the message which was encoded by workstation W1. If, after decoding the message, the coded number X matches the number generated in the nonce, it follows that the smart card SC knows that workstation W1 is indeed who it says it is. The smart card SC may then sign a delegation certificate D1' that says the entity having private key corresponding to public key W1delP is authorized to speak for P. This is illustrated in FIG. 7C.

Similarly, if a second workstation is needed to complete the chain, a second new delegation pair K2 (W2delS, W2delP) is generated by W2, and another delegation certificate D2' is generated and signed with the new private delegation key of workstation W1. This second delegation certificate D2' says that "the entity possessing the private key corresponding to public key W1delP says that the entity possessing the private key that corresponds to the public key W2delP may speak on its behalf."

The delegation certificates D1 and D2 continue to include the workstation names W1 and W2 because those names are needed for lookup in the access control lists (discussed below).

The system resource Ws only accepts requests from W2 on behalf of the user P if the requests are signed by the new delegation key W2delS. This is because the delegation certificate D2 (from W1 to W2) does not totally delegate authority to workstation W2. The delegation certificate D2 only delegates authority as long as workstation W2 possess the secret key W2delS.

In this embodiment of the invention, when the user P wishes to end a session he can instruct workstation W1 to erase the new private delegation key, W1delS. Because the private key W1delS was held secret by workstation W1, the delegation chain is completely destroyed, because any principal that attempts to use the delegation certificate must demonstrate knowledge of the secret key W1delS. Since there was only one copy of this secret key (and it has been destroyed), no one can use the delegation certificate D1 to make requests. Thus, even if workstation W1 (or W2 for the second delegation) is subsequently compromised, it cannot make unauthorized requests from P because the key necessary to support the delegation certificate D1 (or D2) has been destroyed.

As will be noted, this embodiment removes the need to place extended trust in workstation W1. In the present invention, workstation W1 needs only to be trusted to operate faithfully while the user is present, and to erase the private key W1delS when the user logs out. Trusting an entity to delete a single key at a certain time requires less trust than trusting that an entity never be compromised after a delegation. This is particularly important in an environment where the entity is a "public workstation" no longer under control of the user once he leaves the vicinity.

(c) Generation of Session Keys. Creating a new public/private key pair is a very computer intensive operation that may be improved in one or more of several ways.

In "pre-generation," a workstation can create several session keys ahead of time as a background process and have several key pairs available for when a user wishes to log in.

"Key pair economy" can be pursued by instructing the workstation to save a session key pair if no external requests are made using that pair, and delete the user's initial delegation certificate. This operation is premised on the fact that if the workstation has made no external requests, then the user's delegation certificate has not been transmitted outside the workstation, and erasing it assures its destruction.

Simplification of session delegation can be sought: if the identity of the workstation is not needed to implement the access control function, then the session delegation can be simplified by (1) leaving out the identity of the workstation in the delegation certificates (e.g., having a certificate say that "P authorizes anyone having private key corresponding to WxdelP to speak on his behalf"; or (2) not generating any additional session keys after the first (e.g., W1 simply transmits a copy of the user's delegation certificate and the new public delegation key to W2). For the embodiments using alternative number 2, a privacy-protected channel must exist between W1 and W2.

9. ACCESS TO OBJECTS

As discussed above, an "object" is something in the system to which access is controlled. Access is controlled for a given object by granting or denying a principal's access to that object in accordance to the security model to be implemented.

Access control decisions for each object are made locally by the reference monitor controlling that object on the basis of specific access control data for that object. The specific access control data for each object are preferably stored with that object in the form of access control lists (ACL).

Generally speaking, when a principal requests access to an object, that object's reference monitor attempts to locate the requesting principal's identity in the ACL for that object. If the principal's identity is found, the access requested by the principal is compared to the access allowed by the ACL entry. If the ACL entry indicates that the access requested is allowed, then access is granted; if the requested access is not allowed, or the principal cannot be found in the ACL, then access is denied.

9.1. Access Control Lists (ACL)

(a) Generally. Generally speaking, an "access control list" (ACL) is a data structure that associates access rights with sets of named principals. In its most basic sense, an ACL may comprise a list of the names of the principals which are allowed access to the object associated with that ACL, and an indication of the access rights that are allowed.

The creation of ACLs is a matter of routine work for a person skilled in the art having the benefit of this disclosure. Past examples of ACLs include the SOGW

protection mask, the ACL in VMS, and the OGW file protection modes in Unix.

The access rights for a particular principal are the operations that a reference monitor will allow that principal to perform on its object. Traditional access rights are read, write, delete, etc.

In accordance with the present invention, access rights include any request that can be implemented by an object. Thus access rights include such things as the ability to make requests of a file service, to join a cluster, and to change a process's priority.

The access rights are generally associated with principals through the use of "access right expressions." Such access right expressions will be discussed in detail below.

(b) The ACL as an Object. Because ACLs may need to be read or changed, they themselves fall within the definition of "objects." As such, each ACL itself has an ACL that specifies who can read or modify it. In order to avoid infinite regress, the second level ACL (i.e., the ACL's ACL) may be its own ACL. Thus, in addition to the access rights discussed above, the ACL for an object may include an entry indicating that the principal can read or write to that ACL.

It may be desirable to grant read access to an ACL to those principals that are able to read the corresponding object, and write access to those having write access to the object. Such an embodiment is illustrated in FIG. 8. In that figure an object 60 (a file ALPHA) is illustrated with its ACL 62. As shown, the principal JOHN has read access to the object 60 and principal ALICE has read access to both the ACL 62 and the object 60. An arrow 65 indicates that the ACL 64 is its own ACL.

(c) Authentication and Principal Storage. To mediate access, the reference monitor for an object must interpret that object's ACL. For security reasons, each principal or group named in the ACL should be authenticated by the reference monitor. (The authentication process is discussed above)

(d) ACL Structure. As discussed above, the ACL is a structure for associating access rights with a set (or sets) of principals. A system in accordance with the present invention uses a variety of operators to construct the sets and to associate the access rights. These operators are discussed in detail below. Development of hardware (or software) implementations of the disclosed operations is routine work for a person skilled in the art having the benefit of this disclosure.

The access rights granted to a particular entry in a group may be controlled through the use of access right expressions. Thus, a group may sometimes be thought of as an mini-ACL.

When an ACL includes a group as an entry, the contents of that group in effect become part of that ACL. As FIG. 9A illustrates, an ACL 70 may contain several principal entries 72, as well as the access rights associated with those entries 74. The ACL 70 may also contain group entries, for example group TEACHERS 75. These group entries effectively make the groups referenced thereby part of ACL 70.

FIG. 9B roughly show the group definition for group TEACHERS. Thus principal BOB (while not directly in the ACL 70) will be allowed access to the object controlled by ACL 70 because BOB is found in group TEACHERS. Because ACL 70 grants only delete-type access to TEACHERS, and therefore to BOB, he cannot write to the object controlled by ACL 70.

9.2. Access-Right Expressions and Constructs

An access right expression specifies the specific access rights granted to a set of principals (or groups). This represents the ultimate purpose of the ACL and is therefore discussed in detail here.

FIG. 10 illustrates the most basic access right expression. In that figure, a principal-set 80 is granted the specific access-right 81 listed. The symbol => represents such an granting of access rights. As will be discussed below, a principal-set is a grouping of principals by either name or group.

FIG. 10 also illustrates a second form for access right expression. As illustrated, symbol > indicates that the principal-set 80 may be delegated the specific access-right 81'. As discussed below, a principal may delegate authority to other principals to act on his behalf. An expression such as the one illustrated in FIG. 10 prevents allowing an untrustworthy system from being delegated too much authority. Delegation is discussed in detail above.

FIG. 10 also illustrates a third form that an access right expression may take. In this expression, the principal-set 82 is listed with no access rights. The listing of a principal-set with no access rights implies that the specified principals are granted all possible access rights.

The principal-sets discussed above may comprise a listing of principal names, a group, or a construct of principals and/or groups. For purposes of this specification, the term "principal-set" will refer to a list of principal names, while the term "NAME" will refer to a group.

When a listing of principal-set comprises a listing of principals, each of the listed principals is granted the access rights indicated by the access right expression. For example, in FIG. 11, principals BOB, JOHN, and CARL are all given the access right S, as are all of the principals in group ONE.

Four principal-set constructs are utilized in the embodiment illustrated here. Each will be discussed separately.

(a) Principal-set OR Principal-Set: The UNION of Groups. The construct A OR B, where A and B are principal-sets, means that A and B are both members of the set. In other words, to be allowed access, the accessor must be either A or B. The following equation illustrates this construct.

A OR B=>S

principal-set=>access-right

The above caption indicates that both A and B have access right S for the object controlled by the ACL having this access equation. If the principal seeking access is either A or B access right S will be granted. This is because the basic principal-set comprises the union of A and B.

(b) Principal-set AND Principal-set: The INTERSECTION of Groups. When the ACL contains the construct A AND B, to be allowed access, the accessor must be both A and B. In the present invention the operator AND represents the intersection of set A and B. The following provides an example.

A AND B=>S

principal-set=>access-right

In order to prevent the ACL from improperly granting access to either A or B, access should be denied unless A and B are both acting in concert. Only then can the ACL be sure that the entity requesting access to the object (having the authority of both A and B) has the access right S.

(c) Principal-set—name: Group EXCLUSION in Hierarchical Groups. In the third construct, principals or groups being subtracted out of the principal-set must be members of the groups to the left of the "—". Thus, before this construct may be applied to a principal-set (or group) it must be determined that the group on the right is a subgroup of the principal-set (group) on the left. This construct means that the members of the final principal-set comprise all the members of the initial principal-set that do not derive their membership by being members in the named subgroup.

The following provides an example of this construct:

$$A-B=>S$$

principal-set = > access-right

(B is a subgroup of A)

Thus, the present interpretation of this construct yield a final principal-set which is equal to all of the principals in A which do not owe their existence in A to being in subgroup B. When determining whether a principal is a member of A—B all of the subgroups of A except B should be searched for that principal, i.e., A—B means skip subgroup B when searching the subgroups of A.

In the context of the present invention this construct provides several advantages. First, as discussed above, a misbehaving or compromised group may assert that a particular principal is not a member of a group (i.e., it is not possible to reliably assert group non-membership).

Under traditional interpretations of A—B (i.e., all of the members of A except those that are members of B) an improper granting of access rights may result. For example, assume group A contains subgroup B, and principal ALICE is a member of subgroup B. (This example is illustrated in FIG. 12). If subgroup B is compromised, it may assert that ALICE is not a member of that subgroup. The traditional interpretation of A—B=>S would improperly grant ALICE access right S since she in not found in subgroup B.

Under the construct of the present invention compromise of subgroup B cannot affect the granting of access right S. This is because the data from subgroup B is simply not used in interpreting this construct; thus, withholding data that will not be used cannot make any difference.

A second advantage of the present construct is that the broadest possible access is allowed for the principal being searched by the ACL. For example assume the groups and subgroups illustrated in FIG. 12. As may be seen, principal BOB is a member of both subgroups B and C. Under the traditional interpretation of A—B=>S, BOB would be denied access right S because he is a member of group B.

Under the method of interpretation of the present invention, however, BOB would be granted the access right because his membership in group A is not solely dependant on his membership is subgroup B. Thus in the present invention subtraction of one group does not affect other group memberships.

(d) Principal-set—C(name): Group EXCLUSION. The fourth construct allows for traditional subtraction. A—C(B) includes all of the members of A except those who are also members of B. In this construct B does not have to be a subgroup of A.

Generally, this construct should not be used where the reference monitor for an object does not have guaranteed access to the entire definition of a subtracted group. Such a case may occur where the subtracted group (B) includes subgroups that are not accessible by the reference monitor. If the reference monitor cannot expand the subtracted group through all its subgroups, there is a danger that improper access will be allowed. To avoid this situation, the reference monitor may deny access in all such situations.

9.3. Compound Principals in an ACL

The ACL may be used to restrict access to a user under certain circumstances.

(a) Limiting Access to Specified Computer Systems. In a distributed system, each system on the network may not be equally trusted. In these situations it may be desirable to limit a users access to certain system resources depending on which system he is operating from.

For example, one workstation may be highly secure, while another is relatively unprotected. In this example, it may be desirable to prevent a used from gaining access to highly sensitive resources through the unprotected workstation, but to allow the same user access when operating from the protected system. Thus, the access rights of the user depend on the system within which he is operating.

In order to implement this security policy, a new principal construct ON may be introduced. Thus, when a user U, makes a request through a workstation W, the workstation makes a request to a system resource (e.g., a server) as U ON W. This request is allowed by the user delegating to the workstation authority to make requests in the form of "U ON W says S". (Delegation is discussed in section 8, above). This request is interpreted as "W says U says S, and W is authorized to speak on U's behalf (evidenced by the delegation certificate)."

When such a request is made, the system resource's (object's) reference monitor checks the ACL for that system. If U ON W is found in the ACL, and the desired access S is allowed, then U, through W, is allowed access to the resource.

If delegation is cascaded through multiple systems, all the systems that are included in the compound principal and must appear in the ACL entry. For example, if U delegates to workstation W1, which delegates to workstation W2, which makes a request of a system resource, the ACL for that resource must allow the requested access for "U ON W1 ON W2." Because the degree of security for a chain is equivalent to the degree of security possessed by the least secure member, the order of delegation is usually not important. Thus, "U ON W1 ON W2" may be considered equivalent to "U ON W2 ON W1."

While single workstations were used in the above examples, any of the principals in the ACL may be groups. For example, an ACL having a principal-set entry of "U ON SECURE" would only allow access to user U if logged into any of the workstations in group SECURE. If this construct is used, and a user is to be allowed access to a system regardless of the delegated

systems involved, the construct "U ON*" may be used. Such access right expressions may allow access to U regardless of the chain of delegation.

(b) Limiting Delegated Access Rights. In certain situations, a user with a high degree of trust may wish to use a computer system, but not to trust that system with all his capabilities. In other words, the user may wish to delegate to a system only a subset of his total access rights.

To allow the user to limit his delegated powers, it is convenient to introduce the concept of "roles." A role may be a certain group of access rights which the user (operating in that role) may exercise. For example, a user operating in the role of STUDENT may be allowed to read certain limited files. The same user, operating in the role of PROFESSOR, may be able not only to read these files, but to modify their contents as well.

When the user U delegates to a workstation W, he can sign a delegation certificate in the form of "W is authorized to speak for U AS R." In this manner, the workstation W will only be allowed to invoke the access rights that U has in role R. Any request from the workstation W to a system resource will be in the form of "(U AS R) ON W says . . . ." To gain access to the system, the ACL for the requested resource must contain an entry granting the requested access to "(U AS R) ON W"; access should be denied if the workstation only includes "U" or "U ON W".

The ACL for a resource may be simplified if a group is defined to represent the role. If this is done the ACL entry may be expressed as "R ON W." The reference monitor for the resource will then verify that U is a member of R, and allow the requested access if R is listed on the ACL.

(c) Protected Subsystems. Several security systems utilize the notion of a "protected subsystem." A protected subsystem is a particular piece of software, sometimes in combination with a particular user. As discussed above, the degree of trust granted to computer systems in the network may vary. Thus, when considering a protected subsystem, the computer system executing the particular software is an important consideration, since correct execution depends on the trustworthiness of the computer and its operating system.

As discussed above, a workstation is authenticated by possession of a private key. When a protected subsystem Q is running on the workstation, the workstation W may authenticate itself, and assert requests as "W AS Q."

When the workstation makes a request, it should do so as both the protected subsystem, and the user from whom the request is made. For example, if a user U runs a protected subsystem Q on a workstation W, and a request is made of the server, the request should be made in the form of (W AS Q) WITH U says . . . this request essentially indicates that "both U ON W AND W AS Q say . . . ." This request will be granted if the ACL for the requested system has a listing for "(W AS Q) WITH U" and the reference monitor for the resource can:

(1) authenticate both U and W; and

(2) verify the certificate of delegation by U to W.

As discussed above in regards to roles, the ACL may be simplified to "Q WITH U" by defining a group Q consisting of all the systems which are authorized to run the particular protected subsystem. Thus "Q WITH U" is equivalent to "Q AND (U ON Q)."

9.4. Access Control Lists and The Group as an Object

The group (used here to mean a simple or complex list of group member identities) may itself be treated as an object and as such has its own access controls (i.e., its own ACL). The group's ACL thus controls who may see the contents of the group and who may affect its membership.

Groups may be stored in any globally accessible storage device (i.e., a storage device available to all on the network). Generally, any storage device that supports groups should allow for the stored groups to be created, deleted, read, displayed, tested, and modified. Performance of the above listed operations should be limited to those principals having the necessary access rights (as indicated by the group's ACL).

The above described embodiments of the present invention are to be considered as illustrative, and not restrictive. For example, the functions described above may be equivalently implemented in hardware or software as convenient. It will be apparent to those of ordinary skill that many variations are possible without departing from the invention, which is set forth in the following claims.

What is claimed is:

1. In a distributed system, a method for defining the access of a user on a specified workstation to a system resource having an access control list, the method comprising the steps of:

(a) generating an access code indicating that the user has delegated access authority to the specified workstation; and

(b) allowing the specified workstation to have access to the system resource when all of the following conditions (1)–(3) are met:

(1) the system resource receives an access request from the specified workstation;

(2) the access control list for the system resource has an entry allowing the specified workstation to have access to the system resource if the user has delegated access authority to the specified workstation; and

(3) the system resource determines that the user has delegated access authority to the specified workstation.

2. In a distributed system, a method for defining the access privileges of a workstation to a system resource for an identified user having specified access privileges, the method comprising the steps of:

(a) generating a role code indicating a subset of the user's access privileges to be delegated to the workstation;

(b) generating an access code representing the user acting according to the role code assigned in step (a);

(c) delegating access privileges to the workstation on the basis of the access code generated in step (b);

(d) generating a second access code representing the workstation acting with the access privileges delegated in step (c); and

(e) allowing the workstation access to the system resource when the code generated in step (d) indicates that user is acting according to the role code assigned in step (a) and the user acting according to that role code has delegated authority to the workstation.

3. In a distributed system, a method for defining access for a user to a system resource, the user having

specified access privileges and running a specified computer program on a specified workstation, the specified workstation having certain access privileges, the method comprising the steps of:

(a) generating a role code representing the execution of the specific computer program on that workstation;

(b) generating an access code representing the user acting in conjunction with the assigned role code; and

(c) permitting the user to have access to the system resource when the access code indicates that the workstation is executing the specified computer program and the user is acting in conjunction with the execution of the specified computer program.

4. In a distributed system, a method for defining the access of a group's members on a specified workstation to a system resource, the method comprising the steps of:

(a) generating an access code representing the group on the specific workstation; and

(b) allowing the specific workstation access to the system resource whenever the access code indicates that a member of the group has delegated its access privileges to the specific workstation.

5. In a distributed system, a method for defining the access rights of a workstation to a system resource given the identity of a particular group having certain access privileges, the method comprising the steps of:

(a) generating a role code indicating a subset of the group's access privileges to be delegated to the workstation;

(b) generating an access code representing the group acting in the role represented by the role code;

(c) delegating access privileges to the workstation on the basis of the generated access code;

(d) generating a second access code representing the workstation acting with the access privileges delegated in step (c); and

(e) allowing the workstation access to the system resource whenever the second access code indicates that a member of the group, acting in the role assigned in step (a) has delegated its access privileges to the workstation.

6. In a distributed system, a method for defining access for a group member to a system resource, the group member having certain access privileges and running a specified computer program on a specified workstation, the workstation having certain access privileges, the method comprising the steps of:

(a) generating a role code representing the execution of the specific piece of software on that workstation;

(b) generating an access code representing the group acting in conjunction with the assigned role represented by the role code; and

(c) allowing the workstation access to the system resource when the access code indicates that it is executing the specified computer program and a member of the group is acting in conjunction with the execution of the specified computer program.

7. In a distributed system, a method for defining access for a user running a specific piece of software on a specific workstation to a system resource, the method comprising the steps of:

(a) generating a role code representing the execution of the specific piece of software on that workstation;

(b) generating a first access code representing the workstation acting in the role code assigned in step (a);

(c) generating a second access code representing the user acting in concert with the workstation and software as defined by the first access code; and

(d) defining access to the system resource on the basis of the second access code.

8. In a distributed system, a method for defining access for a group member running a specific piece of software on a specific workstation to a system resource, the method comprising the steps of:

(a) generating a role code representing the execution of the specific piece of software on that workstation;

(b) generating a first access code representing the workstation acting in the role code assigned in step (a);

(c) generating a second access code representing the group acting in concert with the workstation and software as defined by the first access code; and

(d) defining access to the system resource on the basis of the second access code.

* * * * *

IX. RELATED PROCEEDINGS APPENDIX

There are no related proceedings.